NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

A COMPARISON OF DETECTION AND TRACKING METHODS AS APPLIED TO OPIR OPTICS

by

Michael R. Krueger

December 2014

Thesis Advisor:	Brij N. Agrawal
Co-Advisor:	Jae Jun Kim

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2014	**3. REPORT TYPE AND DATES COVERED** Master’s Thesis	
4. TITLE AND SUBTITLE A COMPARISON OF DETECTION AND TRACKING METHODS AS APPLIED TO OPIR OPTICS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael R. Krueger			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The objective of this research is to investigate and evaluate detection and tracking algorithms suitable for Overhead Persistent InfraRed (OPIR) coverage of moving ground targets. One of the largest hurdles is operating with a low signal-to-noise ratio (SNR) in a cluttered environment. The local contrast method (LCM) and principal component analysis (PCA) detection algorithms will be explored and tested while centroid and correlation tracking algorithms will be discussed. Kalman and alpha-beta filters will be compared and contrasted as viable track prediction techniques. This work will also provide a solid knowledge base for future research on the High Energy Laser (HEL) Beam Control Research Testbed that the Naval Postgraduate School is developing in partnership with Boeing Directed Energy Systems. While they are different applications, both HELs and OPIR share common detection and tracking strategies. Simulation results show that the LCM is superior to PCA. However, the best results are obtained by combining the two. Kalman and alpha-beta filters handle single targets with a constant velocity or acceleration with ease, but advanced tracking methods like the velocity matched filter to provide constraints would provide a more robust solution when performing multiple target tracking.

14. SUBJECT TERMS infrared detection and tracking, local contrast method, principal component analysis, signal-to-noise ratio, Kalman filter, centroid, correlation			**15. NUMBER OF PAGES** 105
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A COMPARISON OF DETECTION AND TRACKING METHODS AS APPLIED TO OPIR OPTICS

Michael R. Krueger
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 2003
M.S., American Military University, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2014

Author:	Michael R. Krueger
Approved by:	Brij N. Agrawal Thesis Advisor
	Jae Jun Kim Thesis Co-Advisor
	Garth V. Hobson Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this research is to investigate and evaluate detection and tracking algorithms suitable for Overhead Persistent InfraRed (OPIR) coverage of moving ground targets. One of the largest hurdles is operating with a low signal-to-noise ratio (SNR) in a cluttered environment. The local contrast method (LCM) and principal component analysis (PCA) detection algorithms will be explored and tested while centroid and correlation tracking algorithms will be discussed. Kalman and alpha-beta filters will be compared and contrasted as viable track prediction techniques. This work will also provide a solid knowledge base for future research on the High Energy Laser (HEL) Beam Control Research Testbed that the Naval Postgraduate School is developing in partnership with Boeing Directed Energy Systems. While they are different applications, both HELs and OPIR share common detection and tracking strategies. Simulation results show that the LCM is superior to PCA. However, the best results are obtained by combining the two. Kalman and alpha-beta filters handle single targets with a constant velocity or acceleration with ease, but advanced tracking methods like the velocity matched filter to provide constraints would provide a more robust solution when performing multiple target tracking.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AO	adaptive optics
AOCoE	Adaptive Optics Center of Excellence
DOD	Department of Defense
FAR	false alarm rate
FPA	focal plane array
HBCRT	high energy laser beam control research testbed
LCM	local contrast method
MHT	multiply hypothesis tracking
MWIR	mid-wave infrared
NRO	Naval Reconnaissance Office
ONR	Office of Naval Research
OPIR	Overhead Persistent InfraRed
PCA	principal component analysis
SNR	signal-to-noise ratio
SOH	Straits of Hormuz
SRDC	Spacecraft Research and Design Center
SWIR	short-wave infrared
VMF	velocity match filter
VNIR	visible and near infrared

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I am extremely grateful and want to thank Distinguished Professor Brij Agrawal for allowing me to be part of his expert team in the AOCoE for the short time I was conducting research. They are all true professionals and go out of their way to provide help and support to students who need it. Instrumental to me were the guidance provided by both Dr. Jae Jun Kim and Maj Matt Allen. Without their direction and overwhelming contribution to my increased proficiency at MATLAB, this work would never have been completed.

I would also like to acknowledge the Navy and the Information Warfare community for recognizing how important education is and providing me the opportunity to pursue my dreams in receiving a bachelors' and two master's degrees while serving my country. I also wish to thank the target tracking engineering team at Boeing Directed Energy Systems and Tian Ma from Sandia National Laboratories for taking the time to explain to me what current experts are working on in this topic.

I am also abundantly appreciative of my family for always giving me the strength, encouragement, and love I needed when stress levels were high. My wife, Meredith, and son, Alden, have been very patient as Daddy was locked away in his "cave." Thank you for your understanding and continued support for my goals.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

The United States Overhead Persistent InfraRed (OPIR) architecture consists of both sensors and ground stations operated by both the Intelligence Community (IC) and Depart of Defense (DOD). OPIR provides persistent or near-continuous infrared coverage from space. These space systems and sensors work together to passively collect energy in the visible and near infrared (VNIR), short-wave infrared (SWIR), and mid-wave infrared (MWIR) bands. Collection is then processed as an event or a scene similar to an image. The OPIR architecture supports multiple mission areas including missile warning, missile defense, technical intelligence and battle-space awareness. The performance of OPIR depends on many system components such as optics and optical materials, focal plane arrays (FPAs), line-of-sight stabilization, and onboard processing. One of the most challenging tasks for further enhancement of the OPIR capabilities is in the development of efficient and accurate detection and tracking algorithms for moving ground targets [1], [2].

Target tracking has been an active research area for many years over a wide area of applications. Target tracking in general is considered a difficult problem due to abrupt object motion, changing appearance patterns of the object and the scene, non-rigid object structures, object-to-object and object-to-scene occlusions, and camera motion. OPIR optics are far away from potential targets and this poses further challenges in target detection and tracking of small moving objects due to limited resolution.

The Adaptive Optics Center of Excellence for National Security (AOCoE) at the Naval Postgraduate School (NPS) has focused on research and development of technologies related to large aperture space telescopes and directed energy systems. In this endeavor, NPS is developing a High Energy Laser (HEL) Beam Control Research Testbed (HBCRT) in partnership with Boeing Directed Energy Systems (DES). For directed energy systems such as the HBCRT, tracking algorithms must operate robustly under highly cluttered environments. The AOCoE has been working on the following tasks and sub-tasks during 2014 to further their goal of conducting research in beam control and adaptive optics technology for maritime deep turbulence environments [3].

1. Investigation of Algorithms Suitable for Detection and Tracking of Moving Ground Targets for OPIR Optics

- Survey of target detection and tracking algorithms including feature based correlation track algorithms
- Identify the target of interest for OPIR
- Identify detection and tracking algorithms suitable for OPIR optics and targets of interest

2. Evaluation of Tracking Algorithms

- Simulate OPIR optics and target tracking scenarios
- Applying candidate target detection and tracking algorithms using simulated sensor data (or using actual sensor data if available), analyze the effect of tracking bandwidth and jitter stabilization on track robustness as a function of target signal-to-noise ratio (SNR) and background clutter
- Evaluate the performance

A unique capability of OPIR that is of particular interest to DOD and the Navy is the ability of sensors to track ships on the water by collecting the reflected radiation off the vessel and its wake. In optimal conditions a ship is moving at a relatively constant velocity and there is minimum cloud cover to hinder detection. This becomes a straightforward linear estimation problem in which standard techniques like the Kalman filter are more than sufficient. However, in an extremely cluttered maritime environment like the Straits of Hormuz (SOH), it can become difficult for OPIR to assign the right track to the right ship. This is where “track before detect” methods like a velocity match filter (VMF) and applications of Munkres algorithm can be helpful.

The objective of this thesis is to investigate and evaluate algorithms suitable for detection and tracking of moving ground targets for OPIR optics and evaluate the effectiveness of these algorithms. The algorithms will be run against a simulated maritime scenario. This research will also provide baseline knowledge and a foundation for further study into suitable detection and tracking algorithms for the HBCRT.

B. THESIS OVERVIEW

Chapter II summarizes current applications utilizing robust detection and tracking including HELs and OPIR. The setup of the HBCRT is also discussed.

Chapter III outlines the realistic naval swarm scenario threat faced in the SOH and how that will be modeled in MATLAB.

Chapter IV explains both the local contrast method (LCM) and principal component analysis (PCA) detection algorithms used in this simulation. Signal-to-noise ratio (SNR) and probability of detection and false alarms are also discussed.

Chapter V summarizes the tracking process with research into centroid and correlation trackers. The Kalman and alpha-beta filters will be analyzed for their ability to predict the track and update position and error estimates.

Chapter VI will present the metrics and data resulting from detection and tracking simulations.

Chapter VII provides concluding remarks and opportunities for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

Acquisition, tracking and pointing (ATP) is a highly critical and evolving technology of high interest to the Navy and U.S. military in general. ATP is the backbone of several military applications including directed energy weapons, infrared search and tracking (IRST) radars aboard ships and aircraft, and space-based imaging and non-imaging systems alike to include OPIR spacecraft.

A. HIGH ENERGY LASERS

The laser was first invented in the 1960s and the possible implications and applications became of great interest to the military. In recent years, the HEL and other directed energy weapons have significantly evolved and are almost ready to be used on the battlefield whether that is on land, in the air, or on the sea. The ability to deliver a weapon at the speed of light without running out of ammunition could significantly enhance military capabilities.

The DOD has developed several HEL weapon systems for different purposes to include the AL-1A Airborne Laser (ABL), U.S. Army Tactical High Energy Laser (THEL), mobile HEL technology demonstrator (HEL TD), and the advanced tactical laser (ATL). These can be seen in Figures 1–4.

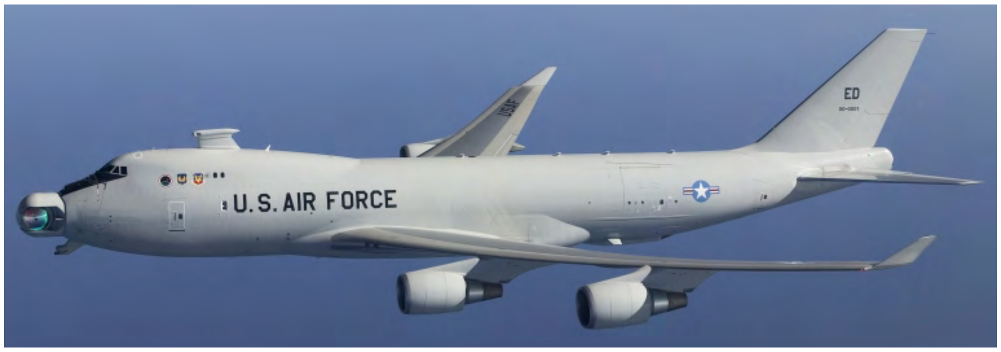

Figure 1. AL-1A ABL prototype in flight (U.S. Air Force) (from [4]).

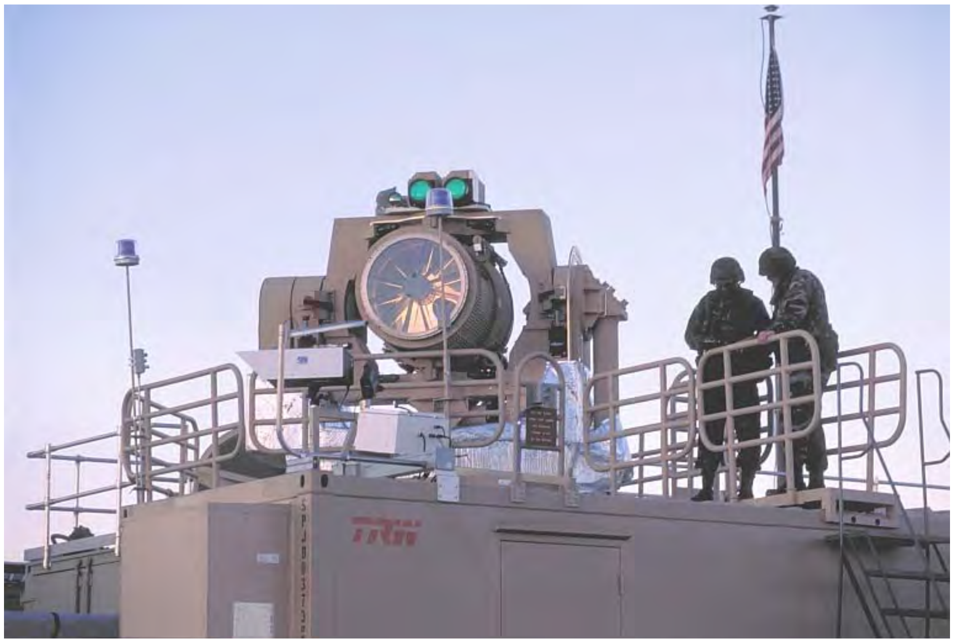

Figure 2. U.S. Army THEL beam director turret (from [4]).

Figure 3. HEL TD (from [4]).

Figure 4. Special Operations Command ATL (from [4]).

The Office of Naval Research (ONR) is developing several HEL weapon systems for the Navy and Marine Corps such as the ship based Solid-State Laser Maturation Program, helicopter based High Energy Fiber Laser Program, Ground Based Aerial Defense, and the Free Electron Laser Program. A HEL weapon system has already been deployed on the USS *Ponce* [5], [6]. These systems will provide naval platforms with highly effective and affordable capabilities against a multitude of threats including surface, air, anti-ship cruise missiles, and swarms of small boats [5].

A HEL weapon system has two main subsystems, the laser and beam control. The focus of NPS research is on beam control that consists of several functions: ATP, aim point maintenance, predictive avoidance, jitter control and correction of atmospheric turbulence using adaptive optics (AO). This thesis focuses on the ATP function, which is the major prerequisite for any HEL system. The beam must be pointed at the right target and be able to track that target efficiently [3].

1. HEL Beam Control Research Testbed

In 2011, the NPS Spacecraft Research and Design Center (SRDC) established the AOCoE sponsored by the ONR, National Reconnaissance Office (NRO), and the Air

Force Research Laboratory (AFRL). One of the priority missions for the AOCoE is to develop and operate a HBCRT in order to validate new technologies, reduce risk to DOD programs, and educate both military and DOD civilian students.

The HBCRT (Figure 5) is a unique testbed that will be used for evaluating a multitude of algorithms, sensors, and beam control concepts. It is designed for a 10kW laser, but will initially operate a low power surrogate high energy laser (SHEL). The HBCRT contains a 30cm aperture on axis HEL telescope with an inertial reference unit (IRU), target illuminator, and acquisition suites mounted to the Gimbal positioner assembly (GPA). The gimbal system is mounted on the angular disturbance simulator (ADS), which is used to simulate large angle ship and aircraft motion, as well as high frequency jitter for testing [3].

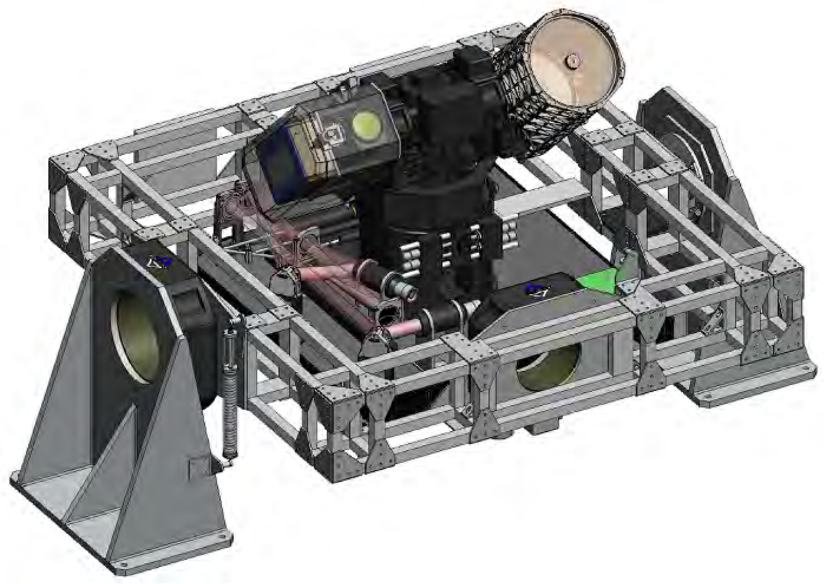
Figure 5. NPS HEL Beam Control Research Testbed.

The HBCRT will utilize a wide-field of view (WFOV) acquisition and tracking sensor (FLIR SC6100 MWIR) as well as a narrow field of view (NFOV) fine track and wavefront sensors (FLIR SC6000 NIR/Xenics Bobcat-640CL SWIR/Xenics Cheetah-640-CL) [1].

The HBCRT will be developed in phases. This thesis will support Phase I, which is the design and development of the ATP, jitter control, and optics control systems. Specific activities under Phase I ATP will be developing capabilities of target cueing, target ranging using a cueing sensor, passive tracking using MWIR sensor, and maintaining pointing stability to less than five microradians (μr) root mean square (rms). Phase II is the integration of these capabilities into the testbed and Phase III will be the upgrade to a 10kW laser [3].

B. SPACE-BASED INFRARED SYSTEMS

Space-based infrared systems typically have scanning sensors like the legacy Defense Support Program (DSP), staring sensors, or a combination of the two. Scanning sensors are in continuous motion and mostly support the missile warning and defense missions. They are looking for any IR event like a missile launch. These are high intensity and short in duration. Staring sensors are focused on one area of interest at a time and can "stare" at that area to capture a scene. This scene can then be processed to detect and/or track targets of interest. OPIR sensors perform these missions using many different wavelength bands. These unique combinations fall into the VNIR, SWIR, or MWIR ranges listed in Table 1 [7].

Table 1. Infrared Radiation Bands Typically Collected from OPIR Sensors.

	VNIR	**SWIR**	**MWIR**
Wavelength (μm)	0.38 – 0.76	0.76 – 3	3 – 8

Infrared systems work much the same way as electro-optical (EO) imagers. A cell in the object space is projected on a detector through both vertical and horizontal scanners. These scanners can eventually place the scene through the detector. As more detectors are added, the SNR increases as does the image quality. These sensors can range from a quad-cell with four detectors to a complex focal plane array (FPA) with any magnitude of detectors depending on the application as seen in Figure 6. Quad-cells are good for high data rates and have very simple data processing but no target features are distinguishable and the output saturates at motions that are half of the target size. With a

FPA, the resolution is much higher resulting in cleaner images in which spatial gates and thresholds can be employed for detection and tracking. However, they are sensitive to noise on any pixel and are much more complex [8].

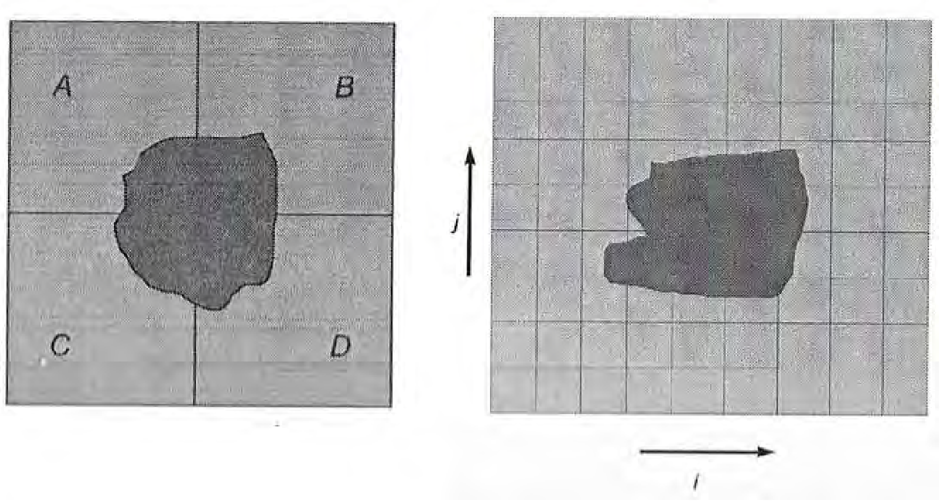

Figure 6. Quad Cell and Focal Plane Array Sensors with Light Spot (from [8]).

The IR sensor data used for this research are images captured in the SWIR band from Landsat. Landsat is a system of satellites in a sun-synchronous orbit that has traditionally been used to image the Earth in support of agriculture, forestry, mapping, geology, water resources, oceanography, and the environment. Therefore, the specific wavelengths of the sensors and ground resolution (GSD) seen in Table 2 are specifically tailored for those missions. Landsat captures images from the visible range through to their thermal infrared sensor (TIRS) bands as it traces the Earth in a push broom pattern. While these are scanning sensors, individual images captured can be used to simulate images that would be processed from a staring sensor with a longer dwell time as in other national OPIR sensors. While the unique qualities like resolution might differ, it is more than sufficient for research of processing techniques like detection and tracking [7].

Table 2. Landsat 8 Operational Land Imager and TIRS Bands (after [9]).

Bands	Wavelength (µm)	Resolution (m)
Band 1 – Coastal Aerosol	0.43 – 0.45	30
Band 2 – Blue	0.45 – 0.51	30
Band 3 – Green	0.53 – 0.59	30
Band 4 – Red	0.64 – 0.67	30
Band 5 – NIR	0.85 – 0.88	30
Band 6 – SWIR 1	1.57 – 1.65	30
Band 7 – SWIR 2	2.11 – 2.29	30
Band 8 – Panchromatic	0.50 – 0.68	15
Band 9 – Cirrus	1.36 – 1.38	30
Band 10 – TIRS 1	10.60 – 11.19	100*(30)
Band 11 – TIRS 2	11.50 – 12.51	100*(30)

The simulation used for this research uses an image of the Straits of Hormuz (SOH) from Landsat Band 6 seen in Figure 7. Most of the Landsat bands have a spatial resolution of 30m with the exception of Panchromatic, which can be acquired at a higher resolution and the TIRS bands which are acquired at 100m resolution, but resampled down to 30m for product purposes [7].

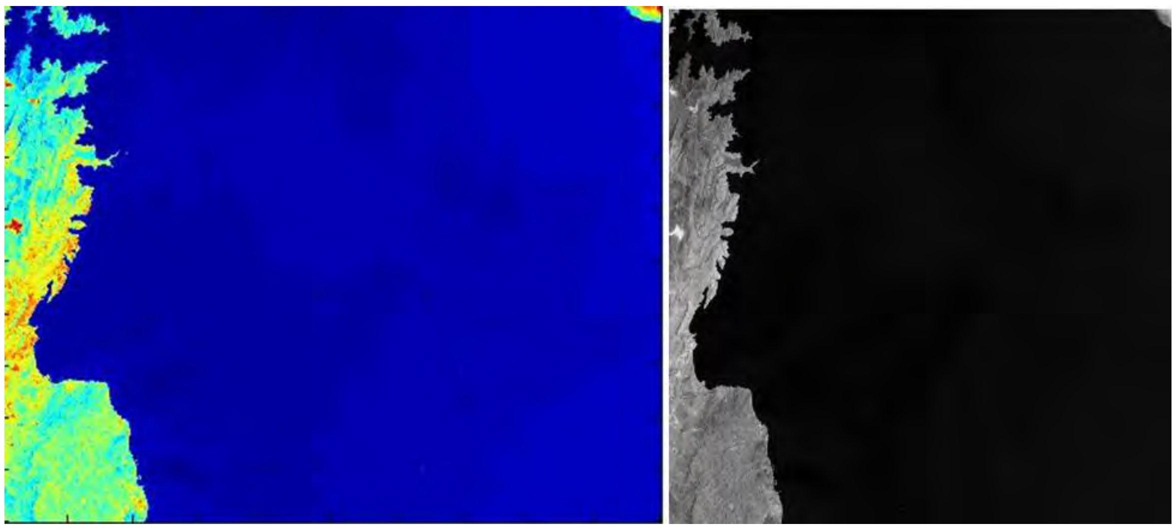
Figure 7. Image of SOH from Landsat in Band 6 and Grayscale.

The detection and tracking problem in this area of the world becomes quite difficult for many reasons. It is a severely cluttered environment which presents many issues when trying to filter out the background noise and track multiple ships that are in very close proximity to each other. In many cases, two small ships might be closer than the GSD of the sensor and it becomes difficult to distinguish separate tracks. Methods for achieving the desired effect will be discussed in following chapters.

III. SCENARIO

A. THE STRAITS OF HORMUZ

The United States Central Command (USCENTCOM) area of operations (AOR) has been and continues to be arguably the most tumultuous AOR in the world. This region comprises 20 countries and is home to an unprecedented number of ethnic groups, languages and traditions. All five of the world's major religions are also practiced. It is also home to almost 60 percent of the world's oil and natural gas reserves which makes it strategically important for the United States. Major sea lines of communication for international trade and commerce pass through the CENTCOM region to include the three critical maritime chokepoints; the Suez Canal, the Bab el-Mandeb Strait, and the Straits of Hormuz. Almost half of the world's oil and natural gas passes through the SOH.

In addition to protecting trade and commerce, one of CENTCOM's top 10 priorities is to combat the counter-maritime and anti-access threats posed by Iran. Iran poses an immense threat to not only commercial maritime traffic, but also U.S. naval forces in the area assigned to United States Naval Forces Central Command (USNAVCENT). The smuggling of weapons and other contraband routinely leaves and arrives in Iran as well. For these reasons, the monitoring of activity in the SOH is a national priority. A key enabler in this mission is space-based intelligence, for which OPIR is an important piece [10].

B. SWARM TACTICS

The U.S. Navy routinely conducts transits of the SOH and is vulnerable to many different threats associated with high traffic density in a narrow shipping lane. One of the largest threats is that posed by fast moving small boats. The Iranian Islamic Revolutionary Guard Corps Navy (IRGCN) possesses numerous small fast attack craft (FAC) and fast inshore attack craft (FIAC) that have been known to harass U.S. and international naval and commercial vessels. Their potential to use swarm tactics to surround a high-value unit (HVU) transiting the straits poses a serious threat to our warships [11]. Naval commanders at sea have a need for near real-time actionable intelligence in order to act on these possible threats in a normally short time window. The persistent coverage and ability to track moving vessels from the reflected radiation off their hull and wakes makes OPIR a viable candidate to provide the necessary intelligence to these commanders.

C. SIMULATED SENSOR DATA

The SWIR image in Figure 7 is used in this simulation as the original background image. This is a snapshot from Landsat 8 that has been decimated from its original resolution of 30m to 100m/pixel and rotated to orient North and South with the vertical. Ship tracks are then artificially added to the scene along with Gaussian white noise to both the image and detector measurement values. While reality reflects more than merely white noise, the simulated noise is sufficient for evaluation purposes. The simulated motion will be aggregated to generate frames. The simulation also allows for the frame rate to be varied by choosing the number of samples over the simulation time. A frame rate of 0.2 samples/sec (Hz) is chosen for all scenarios. The simulated sensor data will be analyzed with both the LCM and PCA detection algorithms to measure detection rates, precision, and false alarms. These metrics will be presented as functions of signal-to-noise ratio (SNR) and threshold settings. The thresholds will be calculated using the Otsu method and by changing gain values. A simple proof of concept tracking algorithm using centroid with a prediction routine will also be examined. A standard Kalman filter and an alpha-beta (α-β) filter will be compared as prediction methods. Other advanced tracking techniques will also be discussed and their possible applicability evaluated including using a velocity match filter (VMF) and other constraints to aid in multi-hypothesis tracking (MHT).

D. MODELING TARGET TRACKS

Scenarios of varying complexity have been created to compare the detection and tracking methods above. The culminating scenario will be one simulating the swarm attack discussed in Section B. The following equations detail how the target tracks were dynamically modeled and inserted into the scene.

A 5th order polynomial is used to approximate the dynamics of the attacking ships. The equations of motion for a single attacker are shown in Equations 3.1 through 3.3. The time domain is mapped to the τ domain using Equations 3.4 and 3.5. The boundary conditions in the time domain are equal to the boundary conditions in the τ domain, which allows the equations of motion to be approximated using polynomials in the τ domain. By modeling the system in the τ domain the velocity is decoupled from time, and the system can be optimized for time. The polynomial system of equations is given in Equation 3.6. The polynomial approximation for the time derivative of τ is given in Equation 3.7. The polynomial coefficients are found using Equation 3.8.

$$\dot{x} = \upsilon \cos(\psi) \tag{3.1}$$

$$\dot{y} = \upsilon \sin(\psi) \tag{3.2}$$

$$\dot{\psi} = u \tag{3.3}$$

$$\dot{x} = \frac{dx}{dt}\frac{d\tau}{dt} \triangleq x'(\tau)\lambda(\tau) \tag{3.4}$$

$$\dot{y} = \frac{dy}{dt}\frac{d\tau}{dt} \triangleq y'(\tau)\lambda(\tau) \tag{3.5}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & \tau_f^1 & \tau_f^2 & \tau_f^3 & \tau_f^4 & \tau_f^5 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 & 4\tau_f^3 & 5\tau_f^4 \\ 0 & 0 & 2 & 6\tau_f & 12\tau_f^2 & 20\tau_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_0' \\ x_0'' \\ x_f \\ x_f' \\ x_f'' \end{bmatrix} \tag{3.6}$$

$$\frac{d\tau}{dt} = \lambda(\tau) = c_1(\tau) + c_2 \tag{3.7}$$

$$a = A^{-1}x = \begin{bmatrix} x_0 \\ x_0' \\ \dfrac{x_0'}{2} \\ \dfrac{-(20x_0 - 20x_f + 12\tau_f x_0' + 8\tau_f x_f' + 3\tau_f^2 x_0'' - \tau_f^2 x_f'')}{2\tau_f^3} \\ \dfrac{(30x_0 - 30x_f + 16\tau_f x_0' + 14\tau_f x_f' + 3\tau_f^2 x_0'' - 2\tau_f^2 x_f'')}{2\tau_f^4} \\ \dfrac{-(12x_0 - 12x_f + 6\tau_f x_0' + 6\tau_f x_f' + \tau_f^2 x_0'' - \tau_f^2 x_f'')}{2\tau_f^5} \end{bmatrix} \tag{3.8}$$

The 5th order model was implemented with the initial and final acceleration values equal to zero. The maximum acceleration was set and the cost function was updated (Equation 3.9) to incorporate a penalty for maximum acceleration. The penalty cost function constants for acceleration were tuned to return a physically feasible trajectory that met the initial and final boundary conditions (position and velocity).

$$J = t_f + \frac{(\dot{\psi}_{\max} - \hat{\dot{\psi}}_{\max})^2}{\hat{\dot{\psi}}^2} + \frac{(\upsilon_{\min} - \hat{\upsilon}_{\min})^2}{\hat{\upsilon}_{\min}^2} + 10^6 \frac{(\upsilon_{\max} - \hat{\upsilon}_{\max})^2}{\hat{\upsilon}_{\max}^2} + 10^6 \frac{(a_{\max} - \hat{a}_{\max})^2}{\hat{a}_{\max}^2} \quad (3.9)$$

To generate multiple attack trajectories that hit the target at the same time the optimization variable is final time (time domain) instead of τ domain. The τ domain and time domain are related using Equations 3.10 and 3.11. The optimization was solved using the MATLAB function *fminsearch.m*.

$$\tau_f = c_2 t_f \,, \, for \lambda_0 = \lambda_f \quad (3.10)$$

$$\tau_f = \frac{c_2 e^{c_1 t_f} - c_2}{c_1} \quad (3.11)$$

The simulation was developed to generate "*n*" random trajectories of different starting positions. A for loop is used to generate multiple attack vehicles and the cost function is the sum of the cost function for an individual attacker. The final velocity constraint penalty in the cost function allows the final velocity and acceleration to vary, but constrains the trajectories between the minimum and maximum velocity and acceleration.

IV. DETECTION

The first step in an efficient detection and tracking system is conducting pre-detection. There are numerous methods of pre-detecting a signal emanating from within a noisy scene, two of which will be discussed in detail. The LCM involves enhancing the signal while suppressing clutter. PCA uses a pre-determined number of frames to identify static pixels and establish a background image. This is then subtracted from the overall image leaving moving targets in a foreground image. The second step is the detection phase in which targets are identified through statistical thresholding and target size. Thresholds will be determined using both the Otsu method and simple analysis of the mean and standard deviation of the image. Using the threshold value to separate pixels into a binary image, detections can easily be made by analyzing locations where pixels are grouped together. A bounding box is chosen based on target size and any detections that meet the threshold criteria and fall within this gate will be identified. Experimental comparison of the pre-detection and thresholding methods will be provided in Chapter VI. In all detection methods, the key is to find the appropriate SNR that balances probability of detection with number of false alarms.

A. SIGNAL-TO-NOISE RATIO

The successful detection of a target, regardless of sensor, all comes down to the ability to process the signal and filter out noise. The signal power of the target on the sensor compared to the total noise equivalent power (NEP) will affect image quality, probability of detection, and false-alarm rate (FAR) which also depends on threshold setting. For purposes of simplifying calculations, the target is assumed to be a point source. The energy from this source first reaches the system aperture and its irradiance at that entrance is given by Equation 4.1 as a function of radiation intensity in W/unit solid angle and the range to the sensor in cm.

$$H_t = \frac{J_t}{R^2} \tag{4.1}$$

The total signal power, *S*, is found by multiplying the irradiance by the aperture area. This does not take into account any atmospheric losses which will need to be included for detailed calculations.

$$S = \frac{\pi D^2}{4} * \frac{J_t}{R^2} \tag{4.2}$$

Next, each detector must be examined to determine the total NEP at the sensor. The area of each detector is given as a function of instantaneous field-of-view (IFOV), *α*, and focal length by Equation 4.3. The focal length is given in Equation 4.4 where *f/#* is the *f-number*.

$$A_d = \alpha_d f_1^2 \tag{4.3}$$

$$f_1 = (f / \#)D \tag{4.4}$$

$$A_d = \alpha_d [(f / \#)D]^2 \tag{4.5}$$

The noise level at each detector relates the area along with the frequency bandwidth of the received signal in Hz and detector detectivity, *D**, as seen in Equation 4.6. The detectivity is based on the specific type of detector and observed wavelength pulled from the chart in Figure 8.

$$NEP = (A_d \Delta f)^{1/2} / D* \tag{4.6}$$

The received frequency bandwidth takes a little more analysis and requires use of a Fourier transform based on the inverse of the on-target time.

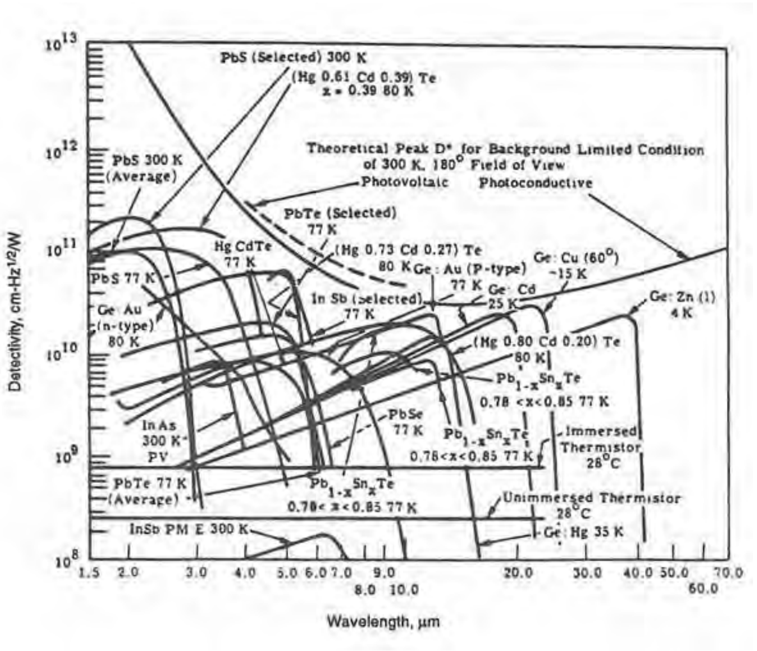

Figure 8. Spectral *D** for commercially available detectors (from [7]).

For a single detector, the number of resolution elements or pixels covered is simply the total sensor coverage over the IFOV in steradians (sr). The on-target time is then simply the frame time over the number of pixels. In an FPA, there are "*n*" detectors working in parallel. The number of detectors is equal to total solid angle coverage of the sensor, Ω_s, over the IFOV of each detector. The scanning efficiency, K_n, is also taken into consideration here.

$$n_r = \frac{\Omega_s}{\alpha_d} \tag{4.7}$$

$$t_d = T_f / n_r \tag{4.8}$$

$$t_d = \frac{n\alpha_d T_f}{\Omega_s} K_n \tag{4.9}$$

The Fourier transform of a step function is sinc (x). For a step function of time t_d, the 3 dB bandwidth is simply $1/t_d$ with the null-to-null bandwidth $2/t_d$. Therefore, the filter noise bandwidth is given by Equation 4.10, where K_{fd} is the bandwidth proportionality constant.

$$\Delta f = \frac{\Omega_s K_{fd}}{n\alpha_d T_f K_n} \tag{4.10}$$

Now, Equation 4.10 can be substituted back into Equation 4.6 to give us the total NEP. Equation 4.12 gives the final SNR calculation for a point source target. A total loss factor, *L*, is added to the final SNR equation that includes the following losses: optics (K_o), electronics (K_e), detector (K_d), monitor (K_m), scanning loss factor (K_n), and bandwidth proportionality constant (K_{fd}) [7].

$$L = K_o K_e K_d K_m (K_n / K_{fd})^{1/2} \tag{4.11}$$

$$\frac{S}{N} = \frac{\pi D J_t D^* L}{4(f/\#)R^2}\left(\frac{nT_f}{\Omega_s}\right)^{1/2} \tag{4.12}$$

The above equations show how the SNR of the sensor attached to an optics payload are calculated. The SNR of the scene or image must also be taken into consideration. For the scenarios in this research, the signal power is that of the targets artificially implanted while the noise will be modeled as white noise with a Gaussian distribution. The SNR of the image is the ratio of the average intensity of the target (ship) to the standard deviation of the noise (background) as seen in Equation 4.13. Each simulation has calculated the variance needed in the background to achieve a desired SNR and then inserted Gaussian white noise with that variance into the image. In this way, different scenarios with differing SNR values can be compared.

$$SNR = 20\log_{10}\left(\frac{\mu_{tgt}}{\sigma_{bg}}\right) \tag{4.13}$$

B. FALSE ALARMS AND PROBABILITY OF DETECTION

Once the SNR for a given target is known, the two important pieces of information from Figure 9 are the probability of detection and probability of a false alarm, P_n.

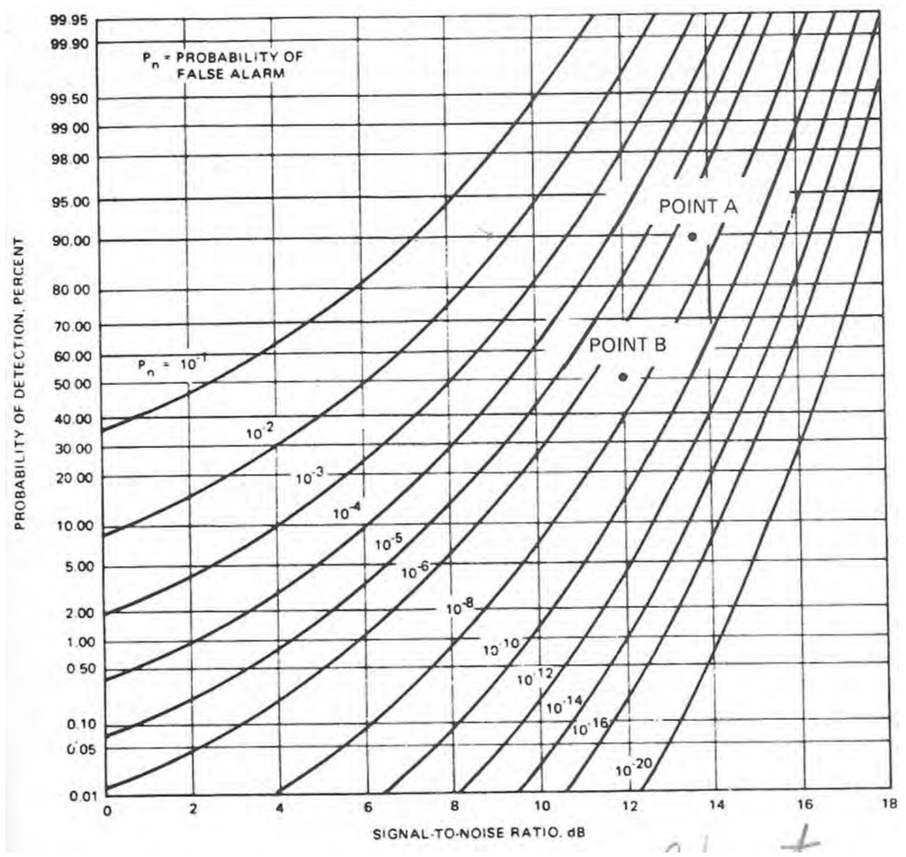

Figure 9. Probability of Detection and False Alarms versus SNR (from [7]).

During each "on-target" time, the FPA of detectors searches for a signal. If there is no signal but noise crosses a certain threshold, false alarms could be triggered. The FAR in false alarms per second is a function of P_n seen in Equation 4.14. Threshold settings will be discussed in the next section, but a given threshold setting is the 50 percent probability of detection at a given P_n [7].

$$FAR = \frac{P_n \Omega_s}{\alpha_d T_f} K_n \tag{4.14}$$

C. LOCAL CONTRAST METHOD

Inspired by the human vision system (HVS) and the derived kernel model (DK), Chen *et al.* [12] propose the LCM for robust small target detection in a low SNR environment. The LCM spawns from the important theory that the gray value of a target pixel will be higher than that of its immediate background and the local contrast, which is the most important feature to the HVS, will be higher than in its surrounding neighborhood. Figure 10 denotes the entire image frame *w*, the local background *v*, and the target region *u*. Region *v* can move around on *w* and *u* can move anywhere along *v*. Furthermore, the target region is broken down into nine cells with the central one labeled 0. Each cell contains *n* x *n* pixels.

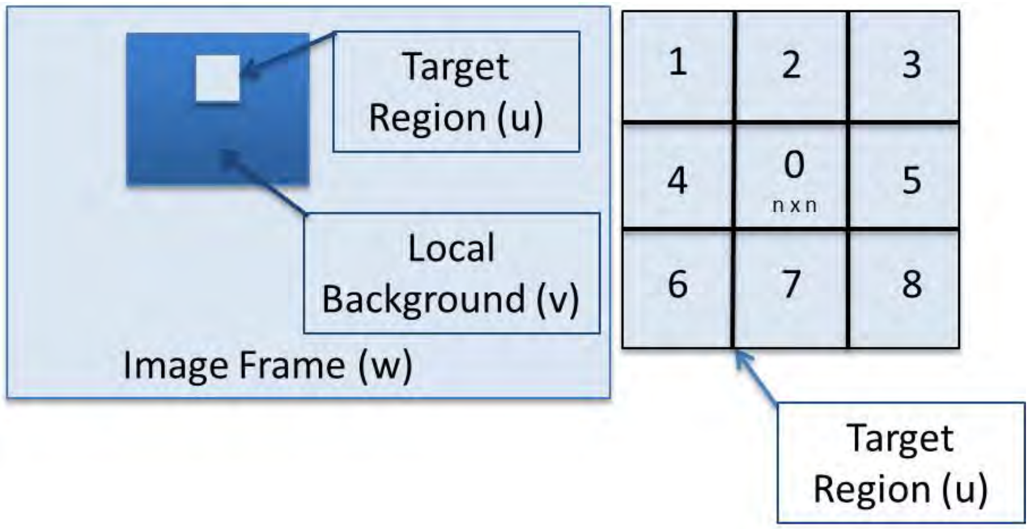

Figure 10. Image patch surrounded by its local background (after [12]).

The LCM algorithm begins by taking the current region and determining the maximum gray level in the central cell, L_n. For each of the remaining eight cells, the gray mean of the *i*th cell is determined from Equation 4.15 where N_u is the number of pixels in the *i*th cell and I_j^i is the gray level of the *j*th pixel in that cell.

$$m_i = \frac{1}{N_u}\sum_{j=1}^{N_u} I_j^i \tag{4.15}$$

The local contrast between the central cell and each surrounding cell is determined from Equation 4.16 and the target is effectively enhanced by gathering each local contrast and replacing the central pixel with contrast map C_n from Equation 4.17.

$$c_i^n = \frac{L_n}{m_i} \tag{4.16}$$

$$C_n = \min_i (L_n^2 / m_i) \tag{4.17}$$

The above LCM shows that if the current region is where the target is located, than L_n will be larger than m_i and the local contrast will be larger than the local maximum. In this way, the target is effectively enhanced. On the other hand, if the current region is in the background, the opposite may be true and the local contrast will be smaller than the local maximum. This will effectively suppress the background. LCM accomplishes both of these simultaneously.

The final step is to detect targets by separating them from the background according to a threshold, T. Chen *et al.* have used a method that effectively takes the mean of the final contrast map and scales it by a factor (usually three to five) of the standard deviation as seen in Equation 4.18 [12]. The Otsu method has also been used heavily and will be discussed further.

$$T = \overline{I^c} + \kappa \times \sigma_{I^c} \tag{4.18}$$

1. Otsu Thresholding Method

The Otsu method differs from the previous thresholding method in that it provides an optimal global threshold for the image. Therefore, a gray-level histogram must first be created. The image pixels are divided into two classes and gray level probability distributions computed. The optimal threshold is chosen by maximizing the between-class variance while minimizing the within-class variance [13]. The Otsu threshold can be computed using the "*graythresh.m*" function in MATLAB.

Overall, the LCM does very well at enhancing weak signals or signals in a low SNR environment. This is potentially valuable to OPIR sensors which are staring at a cluttered scene across large distances. However, by increasing the signals and probability of detection the number of false alarms also increases. This is due, in part, to bright static objects incorrectly identified as targets. PCA is a pre-detection method that can alleviate this issue by filtering out static objects into the background.

D. PRINCIPAL COMPONENT ANALYSIS

The second target detection method examined in this research involves PCA for background subtraction as analyzed by Guyon *et al.* [14]. The original PCA model distinguishes moving objects from the static scene. This is done by subtracting out the "background" to create a "foreground" that shows only those moving objects. The PCA model relies mostly on simple matrix algebra. An *m x n* matrix *A* is constructed where *m* is the total amount of pixels in the image and *n* is the number of frames. Therefore, the rows of *A* correspond to a single pixel and its development over time. The matrix is then decomposed using singular value decomposition (SVD) to come up with the products from Equation 4.19 containing only the principal components of *A*.

$$A_{mxn} = U_{mxm} S_{mxn} V_{nxn}^{T} \tag{4.19}$$

SVD is accomplished by finding the eigenvalues and eigenvectors of AA^T and A^TA. Matrices U and V^T are orthogonal and contain the left and right singular vectors of *A* respectively. Matrix *S* is a diagonal matrix containing the singular values.

A small number, *k*, of principal components is chosen. This simulation uses a value of two. The background, *Bg,* is then calculated from Equation 4.21 where *v* is the current frame. In addition, a desired number of reference frames is chosen. The algorithm will step through the total number of frames in increments of this value to determine what is static. Finally, the foreground, *Fg*, is calculated by thresholding the difference between the current frame and the background image.

$$U_{ij} = U_{1,ij}, 1 \le j \le k \tag{4.20}$$

$$Bg = UU^T v \tag{4.21}$$

$$Fg = \left| v - Bg \right| < T \tag{4.22}$$

The implementation and comparison of this algorithm to the LCM will be discussed in Chapter VI. While PCA provides a robust model of the probability distribution function (PDF) of the background, it does have a couple drawbacks. A limitation is the required size of the target. It must be small and moving. If the target remains in relatively the same position for a long period of time, the algorithm will have difficulty distinguishing it from the background. A possible solution to this problem could be to increase the number of reference frames to give more time for the target to move. However, especially with space-based IR sensors, the total number of collected frames could be an issue if the sensor moves frequently. The PCA algorithm is also mostly limited to gray-scale images. This does not pose a significant issue for IR images, but could have negative impacts to other imaging methods [14].

THIS PAGE INTENTIONALLY LEFT BLANK

V. TRACKING ALGORITHMS

There are many different algorithms used for target tracking, but they all implement the same four basic steps:

- Track initiation
- Data association
- Track smoothing
- Track maintenance

Track initiation is the first step in the process. Any new detection that is not associated with an existing track is used to create a new track. When the algorithm begins, all detections start a new track. Most algorithms keep these tracks in an "unconfirmed" state until they can be corroborated by follow-on updates to that track. A typical standard is to use the "M-of-N rule," which says that for N updates, M hits must be associated before a track can be "confirmed." While values of M=3 and N=5 may be heavily used, the balance for any application is probability of detection and false alarm rate.

Next, the algorithm has to decide which detections belong to which tracks in each frame. This step can be simple if tracking one target, but can become complicated very quickly with multiple targets. All tracks are first updated based solely on predicted estimates from the model being used, usually constant velocity or constant acceleration. A border gate is placed around the track and various methods are used to associate detections to that track. This can be done by associating a target that is closest to the gate, one that exceeds a threshold intensity value within the gate, one that correlates to a previously known target, and/or statistical approaches. A cost function approach is used to choose the optimal association. The distance between the detection and its probable association to every track is directly related to its cost. The Hungarian or Munkres' algorithm then determines the optimal cost for each detection-to-track assignment [15].

Track smoothing involves predicting the next location for each track and updating position and error estimates. There are many techniques for accomplishing this with the proven and generally accepted method being the Kalman filter. The alpha-beta filter will also be explored as a potential candidate for track prediction.

The final step in a typical tracking algorithm is defining when to terminate a track. The same "M-of-N rule" can be applied as before where a track is terminated if the target has not been identified for M out of N opportunities [15]. Figure 11 shows the typical flow of a tracking algorithm.

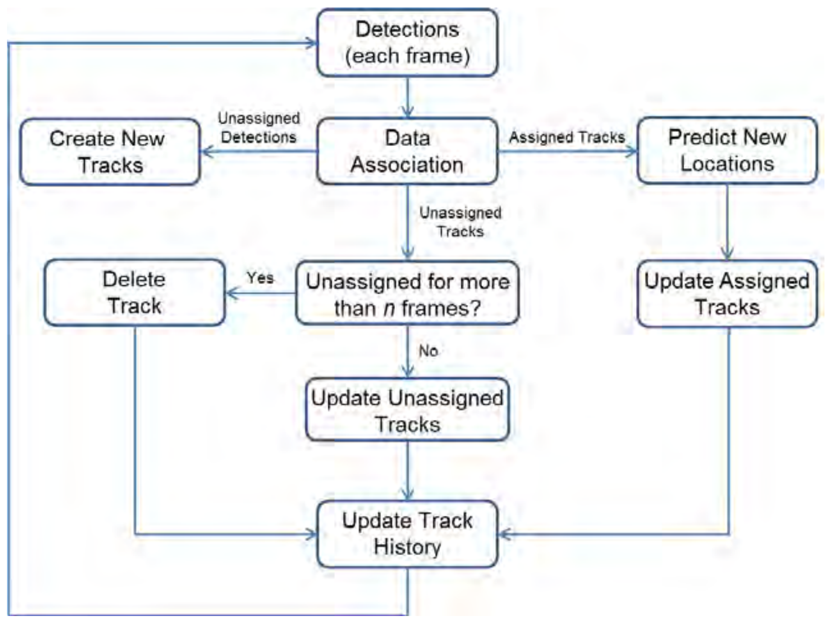

Figure 11. Flow chart for a typical tracking program.

There are three algorithms consistently seen in use across all target tracking applications. Centroid calculates the first moment of the image intensity and then uses thresholds and gating techniques to associate detections to tracks. Correlation trackers maximize the correlation of a reference or "map" within an image frame, and edge tracking algorithms track a target by finding the leading edge of the target with respect to its velocity vector [8]. Edge trackers are simply centroid algorithms with a bias added to the gate. They are extremely useful for missile tracking because the bias forces the gate to become fixed on the centroid of the nose of the missile. In ship tracking, the unbiased centroid is preferred and adequate because they do not travel fast enough to require a bias to compensate. Intuitively, there is not one tracking method that is the best approach all the time. There are advantages and disadvantages of each, but they are all very application specific as with edge tracking. The majority of all OPIR sensors and most HEL applications use a centroid variation tracker because of its simplicity, robustness, and ability to work in almost any system. HELs also use variations of correlation tracking methods that will be discussed further [16].

A. CENTROID

Mathematically speaking, centroid tracking calculates the first moment of the target intensity within the image. A threshold intensity value is determined and applied to the image. This can be done by statistical means or the Otsu method as explained in Chapter IV. The centroid algorithm segments pixels in one of three ways. Binary thresholding makes all pixels above the threshold a 1 and all below a 0. Type I thresholding retains the signal intensity value if it is above the threshold and sets all other pixels to zero. Type II thresholding subtracts the threshold from the signal value for pixels above and sets everything else to 0. Type II is the best method when intensity quantization needs to be considered. Type II is used when the target needs good resolution over the entire image and the dynamic range of the processor ranges from the minimum to the maximum intensity.

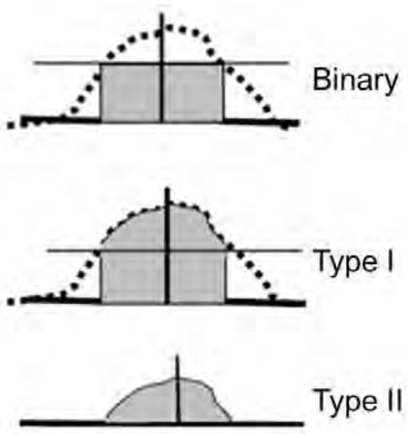

Figure 12. Three thresholding methods (after [8]).

For a quad cell with only four detectors as shown in Figure 6, the centroid can be calculated with Equations 5.1 and 5.2 where *A*, *B*, *C*, and *D* represent the integrated irradiance over that quadrant.

$$\varepsilon_x = \frac{(B+D)-(A+C)}{(A+B+C+D)} \tag{5.1}$$

$$\varepsilon_y = \frac{(A+B)-(C+D)}{\left(A+B+C+D\right)} \tag{5.2}$$

The algorithm also degenerates to Equations 5.1 and 5.2 if the target only covers four pixels. A sensor with a FPA as seen on the right of Figure 6 requires a bit more calculation. Equations 5.3 and 5.4 determine the centroid of a target where *S* is the sensor intensity and I_T is the total summation of differences between the intensity and the threshold value [8].

$$\varepsilon_i = \frac{\sum_i i \sum_j I_T}{I_T} \tag{5.3}$$

$$\varepsilon_j = \frac{\sum_i j \sum_j I_T}{I_T} \tag{5.4}$$

A centroid tracking algorithm will take the currently assigned tracks and perform state estimation to predict the next state of the target. A linear dynamic model is used for targets in these simulations. A bounding box or gate is created based on the specific scenario, target size and speed, and variability. If any detection in the next frame falls into this gate around the predicted location, that detection is assigned to that track. If two or more detections fall into the same gate, another method of association must be used, most likely the closest one to the predicted location is used. The standard Kalman filter and the alpha-beta filter are two of the more heavily used estimation methods.

1. Standard Kalman Filter

The linear state-space model is defined as a four-dimensional vector (Equation 5.5) containing the position and velocities.

$$x_{k+1} = \begin{bmatrix} x(k+1) \\ v_x(k+1) \\ y(k+1) \\ v_y(k+1) \end{bmatrix} = F_x(k) + \upsilon(k) \tag{5.5}$$

The transition matrix, *F*, and disturbance vector, *υ*, are given in Equations 5.6 and 5.7.

$$F = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

$$\upsilon(k) = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} \tag{5.7}$$

A new measurement is taken at each sample and modeled with Equation 5.8 where *H* is the measurement matrix and ω_k is the measurement error vector.

$$z_{k+1} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = Hx_k + \omega_k \tag{5.8}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.9}$$

Equation 5.10 creates the covariance matrix [15].

$$R_k = \mathrm{cov}(\upsilon_k) = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \tag{5.10}$$

The Kalman filter can be initialized many different ways. For purposes of simple comparison, the initial position and velocities are both set to [0 0]. The basic steps of the Kalman filter are to predict the next state and covariance, *P*, with Equations 5.11 and 5.12. Then a measurement is taken (5.13) and the Kalman gain computed (5.14). Finally, the state and covariance are corrected and updated with Equations 5.15 and 5.16.

$$\hat{x}_{k+1,k} = F\hat{x}_k \tag{5.11}$$

$$P_{k+1,k} = FP_kF^T + Q \tag{5.12}$$

$$\tilde{z}_{k+1} = z_{k+1} - H_{k+1}\hat{x}_{k+1,k} \tag{5.13}$$

$$K_{k+1} = P_{k+1,k}H_{k+1}^T[H_{k+1}P_{k+1,k}H_{k+1}^T + R_{k+1}] \tag{5.14}$$

$$\hat{x}_{k+1,k+1} = \hat{x}_{k+1,k} + K_{k+1}[\tilde{z}_{k+1}] \tag{5.15}$$

$$P_{k+1,k+1} = (I - K_{k+1}H_{k+1})P_{k+1,k}(I - K_{k+1}H_{k+1}) + K_{k+1}R_{k+1}K_{k+1}^T \tag{5.16}$$

The filter continually runs and updates its prediction each sample and uses those predictions to initialize the next prediction. In this way, for a linear model the track error is driven down very quickly [17].

The following simulation provides data on the performance of the standard Kalman filter. Each simulation models a single ship traveling from its origin on a 45 degree heading at 20 knots for 100 seconds. With a time delta of 0.1 seconds, this provides 1000 data points. The first two data points are used to initialize the Kalman filter. The standard deviation for range used is 100 yards and one degree for bearing. Each run is looped 100 times as well to obtain the mean error values. Finally, the simulation is run with zero plant noise.

Figure 13 shows the tracking results of the first simulation with no plant noise. The left plot traces the noisy measurements taken at each sample as the ship travels from (500,650) on a heading of 45 degrees. The right plot shows the tracked position according to Kalman prediction estimates. This track clearly follows the true trajectory closer than the measurement. In addition, the error gets smaller as the covariance converges toward the end of the track. This becomes even clearer when computing the mean error over 100 samples as seen in Figure 14. The measurement error remains steady at around 80 yards while the track position error levels out to less than ten yards.

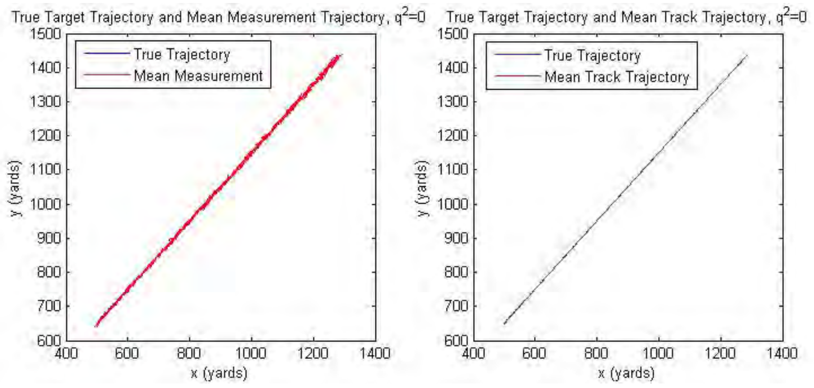

Figure 13. True target trajectory versus measurement and mean track trajectory for q^2=0.

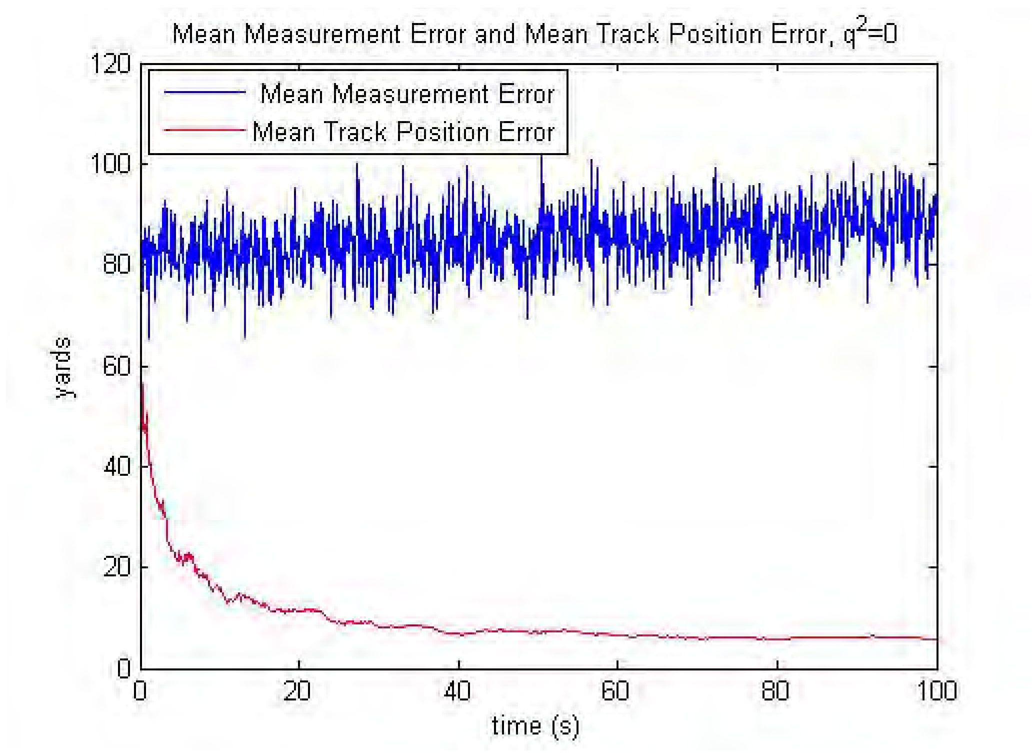

Figure 14. Mean measurement error versus mean track error for q^2=0.

Figure 15 shows the true trajectory (red) and the Kalman error ellipses (blue) based on estimated covariance at regular intervals for two intersecting tracks. It is evident that as the covariance converges the Kalman error gets smaller and smaller. The ellipses are each centered on the position estimate and oriented in the direction of that position from the origin, in this case [400,650]. Figure 16 shows a zoomed in version where error ellipses from the two tracks clearly overlap. In multiple target tracking this poses a data association problem in which some technique for associating those positions with the correct track must be identified.

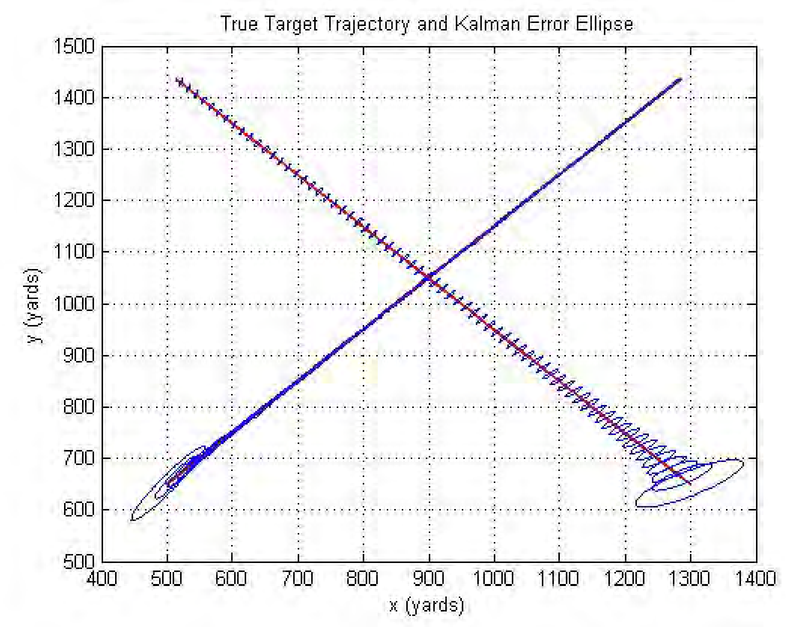

Figure 15. Intersecting tracks shown with Kalman error ellipses.

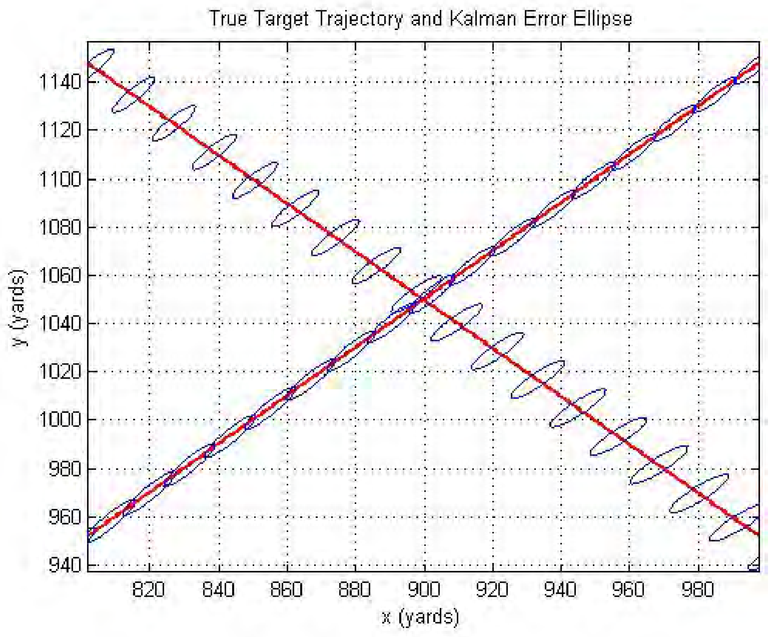

Figure 16. Zoomed in version of Figure 15.

2. Alpha Beta

The Kalman filter can become computationally heavy in a multiple target tracking scenario where each target is estimated independently. However, if the measurement errors for both *x* and *y* are independent then R_k will be a diagonal matrix. This allows the Kalman filter to be broken down into two separate optimal filters to compute the estimated position and velocities separately. These are known as the alpha-beta filters and can save significant amounts of time. Equations 5.17 and 5.18 give the position and velocity equations for the alpha-beta where alpha and beta are the individual gain values [17].

$$\hat{x}_{k+1} = \hat{x}_{k+1,k} + \alpha_{k+1}(z\hat{x}_{k+1} - \hat{x}_{k+1,k}) \tag{5.17}$$

$$\hat{v}_{k+1} = \hat{v}_{k+1,k} + \frac{\beta_{k+1}}{\Delta t}(z\hat{x}_{k+1} - \hat{x}_{k+1,k}) \tag{5.18}$$

The alpha-beta simulation is set up much like the Kalman filter. A ship is placed at the origin and heads on a 45 degree track for 100 seconds and 1000 samples are taken. The root mean square errors are calculated as well and compared to the performance of the Kalman filter. Borges' alpha-beta filter was adapted to run these simulations [18].

The alpha-beta simulations were run three times with varying values for the gain α=0.2, 0.5, and 0.9. Figures 17 through 19 show the position comparisons between the real, measured, tracked, and velocity for those three gain values.

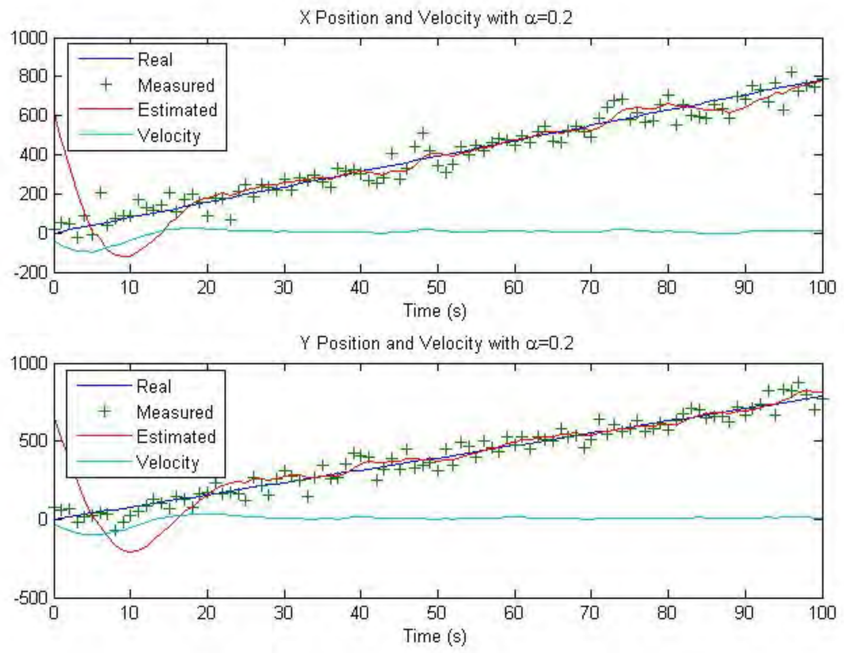

Figure 17. Alpha-beta position comparison for α=0.2.

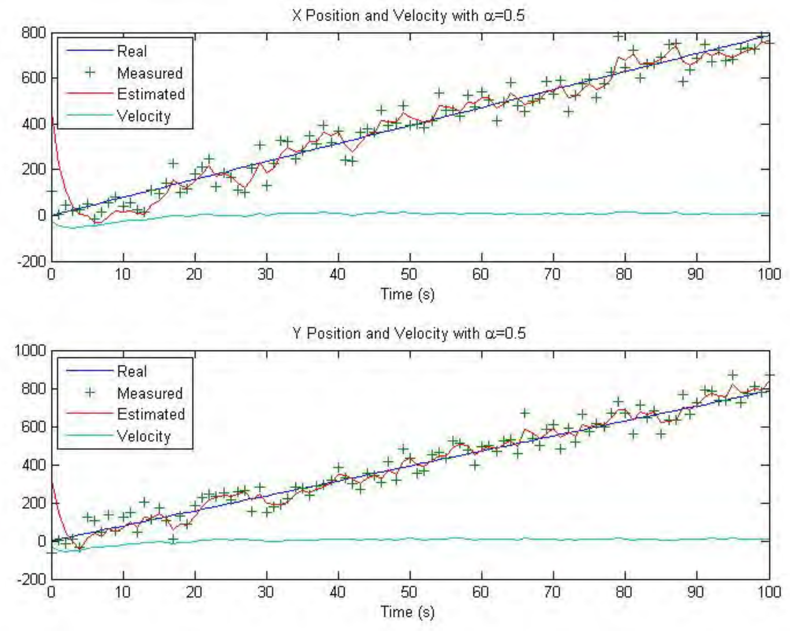

Figure 18. Alpha-beta position comparison for α=0.5.

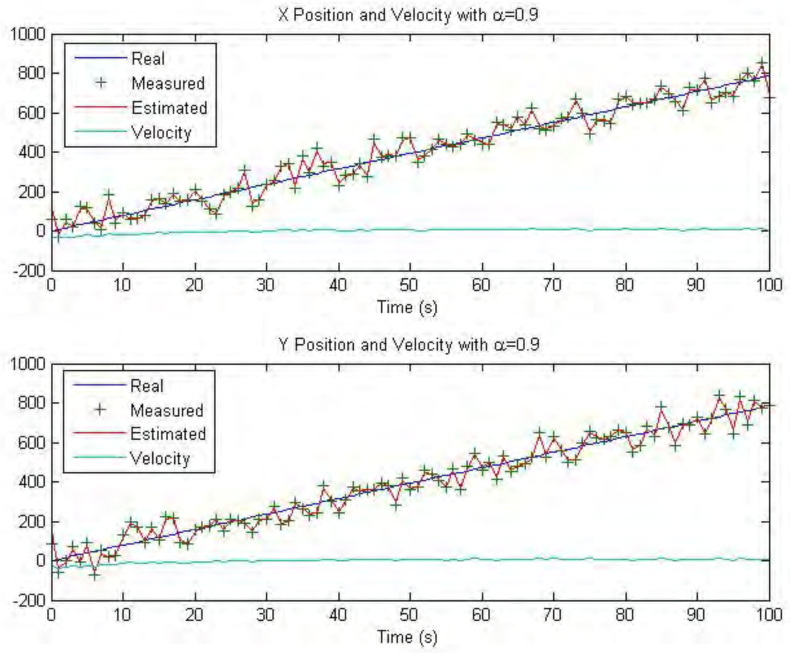

Figure 19. Alpha-beta position comparison for α=0.9.

All three plots show that the estimated trajectory follows very closely with the real track which is to be expected. The higher gain values seem to force the filter down to the smallest error quicker with the lowest gain taking almost 20 seconds to level out. The highest gain value of 0.9 also gave the best mean error over the 100 samples as seen in Figures 20–22.

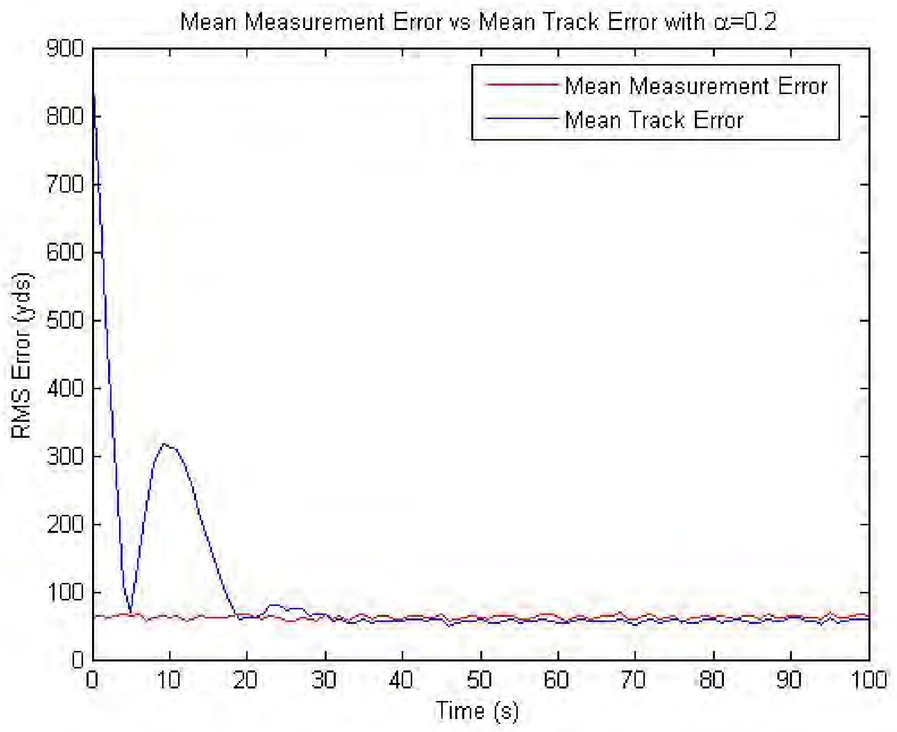

Figure 20. Mean error plotted over time for α=0.2.

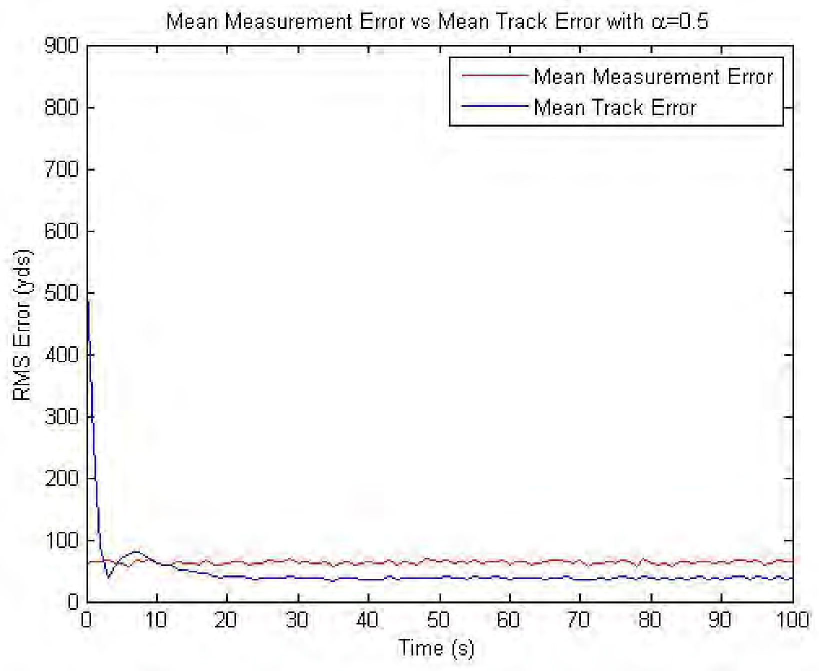

Figure 21. Mean error plotted over time for α=0.5.

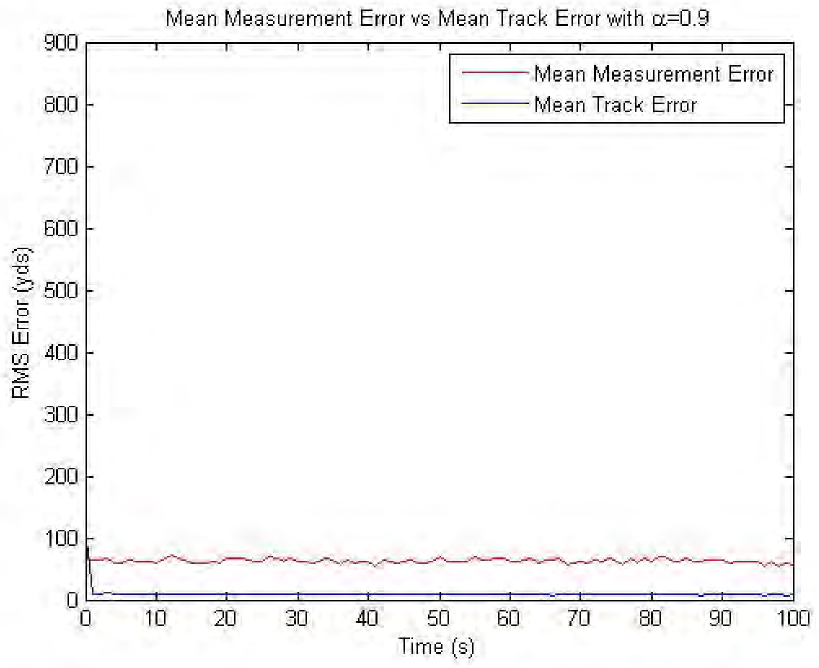

Figure 22. Mean error plotted over time for α=0.9.

Table 3 shows the relative mean measurement and track errors with their associated gain values. The Kalman filter outperformed the alpha-beta with α of 0.2 and 0.5. However, at 0.9 the alpha-beta performed slightly better. This is to be expected since the alpha-beta is simply a de-coupled version of the Kalman running without plant noise against a constant velocity model. If acceleration was involved, the Kalman would most likely produce better results.

Table 3. Mean measurement and track position errors for alpha-beta filter.

	Measurement Error (yds)	**Track Error (yds)**
α=0.2	65	55
α=0.5	60	35
α=0.9	60	10

The other important factor playing into choosing an appropriate filter is computation time. The alpha-beta simulation ran about 13 times faster than the Kalman filter. This code was run on an Intel ® Core ™ i7–4770K with dual 3.50 GHz processors and 16.0 GB of RAM. For a relatively benign computation cutting 13.17 seconds down to 0.90 seconds might not be worth the performance risk. However, when multiple target tracking is needed computational requirements can jump exponentially and this time savings could prove valuable.

B. CORRELATION

Correlation tracking algorithms aim to maximize a cross correlation function, *C*. This function attempts to correlate the image, *S*, to a pre-defined "map" reference, *M*. Equation 5.19 is maximized when the map overlays the target [8].

$$C(x,y)=\frac{\iint S(u,v)M(u+x,v+y)dudv}{\left\{\iint [S(u,v)]^2 dudv \iint [M(u,v)]^2 dudv\right\}^{1/2}} \tag{5.19}$$

When the target shifts so does the maximum of *C*, therefore this can be used as an error signal. The significant drawback to this, as mentioned previously, is the *a priori* target reference needed. This can be an extremely powerful algorithm if the target is known. HEL weapon applications could potentially use variations of a correlation tracker since most potential targets would be known. If not, the NFOV trackers treat most targets as a point source so a sufficient map could be created.

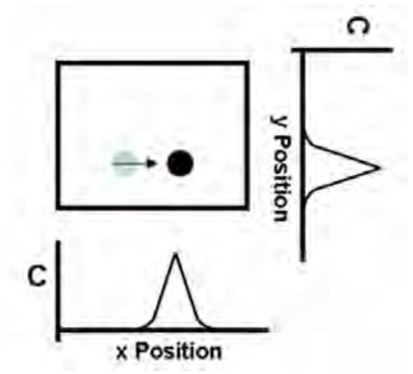

Figure 23. Illustration of the cross correlation function (after [8]).

A summary of the major pros and cons for centroid and correlation trackers is highlighted here [8].

Centroid pros/cons

- Good where target can be segmented from background based on intensity differences
- “Optimal” in least squares sense for parabolic intensity distribution
- Binary is good for targets where glints are an issue
- Threshold type must be determined – Type 1 is best for detection rate, but if considering intensity quantization, Type 2 is best and binary is worst

Correlation pros/cons

- No thresholds or gates are needed although gates are usually still used
- Tracker will still work when target cannot be segmented based on intensity differences
- Rejects clutter moving relative to the target
- There is no “dc reference,” usually initiated with centroid algorithm
- Only sees shifts relative to the reference
- More complex to code and process

VI. DETECTION AND TRACKING PERFORMANCE

A. DETECTION

Three different scenarios of increasing complexity are used to analyze the pre-detection performance of the LCM and PCA algorithms. For each scenario, the LCM and PCA algorithms are tested independently and then in a combined algorithm where LCM is used to first enhance the signal and PCA is run after that to subtract out static objects from the background. The first scenario involves three targets separated by a great distance that are traveling away from each other. This gives a good baseline in how the algorithms will perform under the simplest conditions. The second scenario is one where the three target tracks will cross over each other. This is done to compare how much the detection and precision breaks down when the targets are in close proximity to each other. Finally, the culminating realistic scenario is one where seven targets initially loitering will all converge in a swarm at the same time and place as a high-value unit (HVU). This involves changes in velocity and acceleration of the targets to all hit the HVU at the same time. This is a scenario that the Navy is extremely concerned about. The changes in acceleration and the close proximity of multiple targets all affect the detection performance. The simulations will also be compared against differing SNR values that might be encountered. Noise is added to the image before processing to achieve the desired SNR for comparison. Figure 24 shows the original SOH image and Figure 25 shows a subset image chip with varying SNR values used for the first scenario.

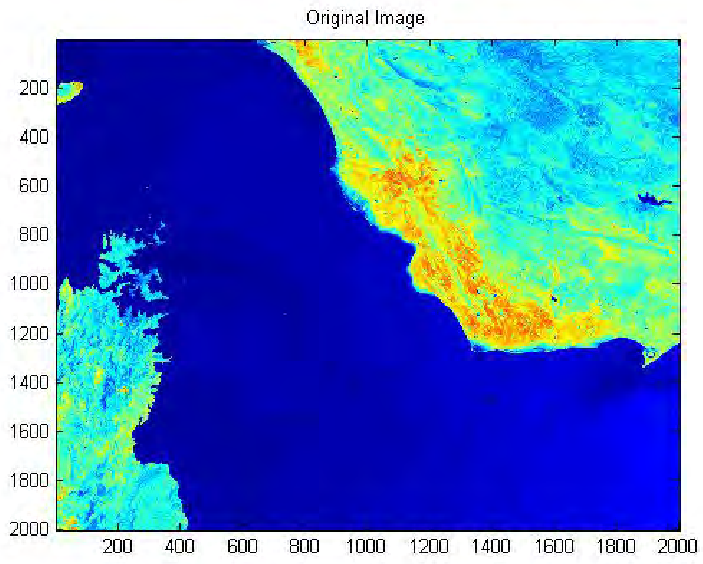

Figure 24. Original infrared image of the Straits of Hormuz.

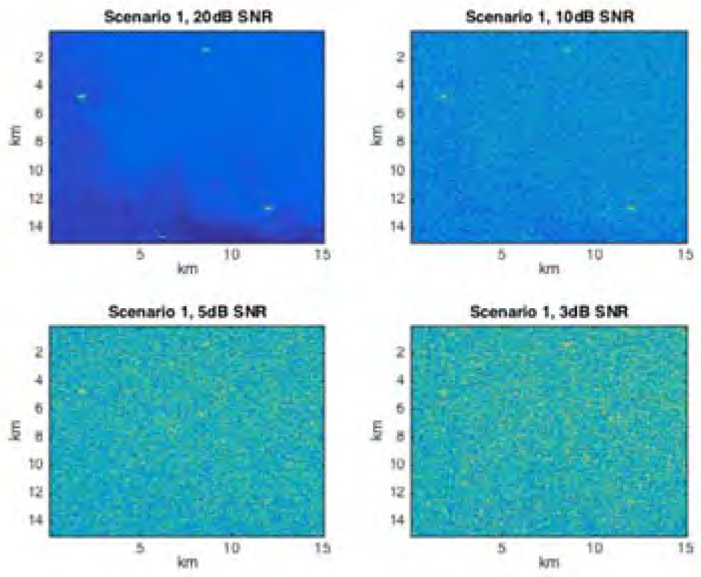

Figure 25. Straits of Hormuz image with SNR of 1, 3, 10, and 20 dB.

Metrics that will be compared are detection rate, precision rate, false alarms, and threshold gain. Equations 6.1 and 6.2 define the detection rate (DR) and precision rate (PR) respectively, where true positive (*TP*) is the number of foreground pixels correctly marked as foreground, false positive (*FP*) is the number of background pixels incorrectly marked as foreground, and false negative (*FN*) is the number of foreground pixels incorrectly marked as background [12].

$$DR = \frac{TP}{TP + FN} \tag{6.1}$$

$$\mathrm{PR} = \frac{TP}{TP + FP} \tag{6.2}$$

The number of false alarms is simply any additional target the algorithm detects that is not one of the inserted tracks. The threshold gain, κ, from Equation 4.17 will be tweaked to determine the setting that provides the best results. All scenarios are run for 300 seconds.

1. Multiple Separated Targets

This scenario consists of three large ships (5x5 pixels) separated by a great distance traveling in different directions that do not intersect. The initial frame of this scenario is once again seen from Figure 26.

Figures 26 to 28 show true trajectories and target detections using LCM, PCA, and LCM then PCA, respectively, for each of the four SNR values being compared.

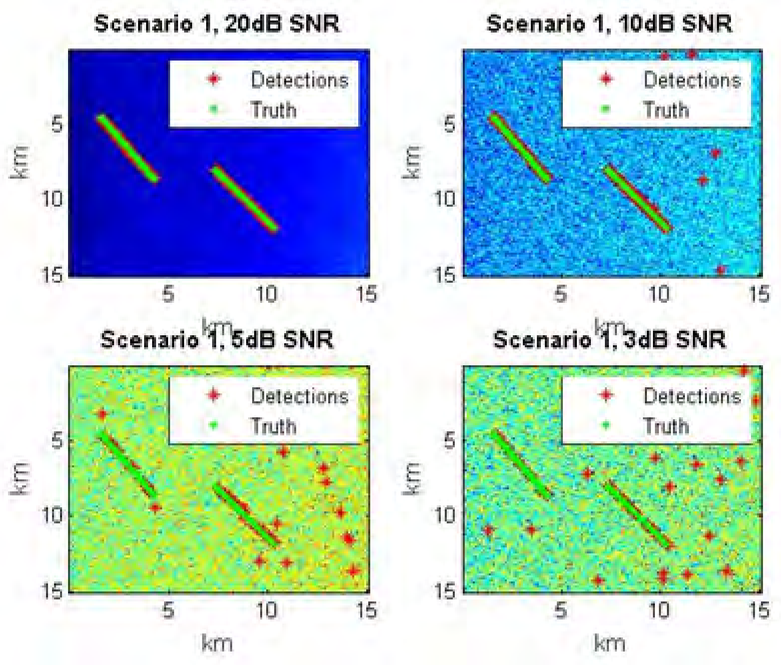

Figure 26. Local contrast method applied to scenario one.

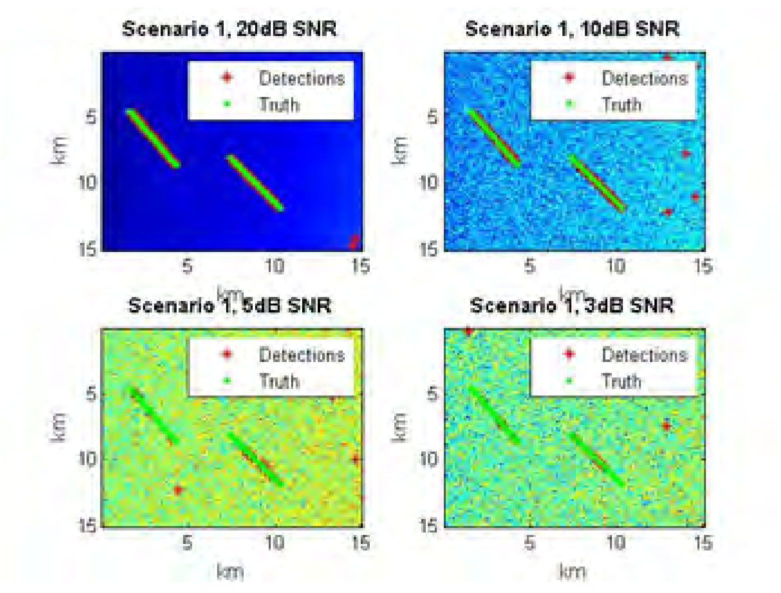

Figure 27. Principal component analysis applied to scenario one.

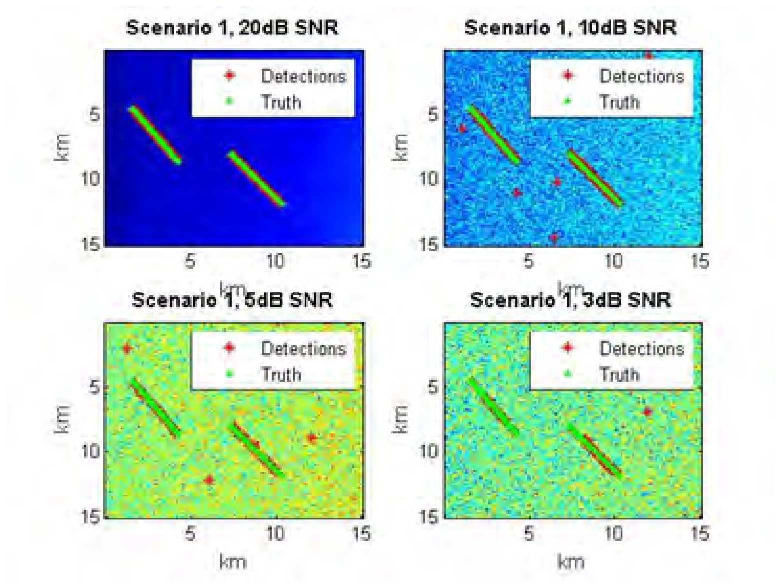

Figure 28. Combination pre-detection algorithm applied to scenario one.

Scenario one provides the expected baseline for algorithm performance. All three simulations get progressively worse as the noise level increases. Figure 27 shows that LCM is able to detect the targets at most points during the track even at the lowest SNR. However, due to the signal boost there are also numerous false detects that result from the boosted noise or from static objects whose intensities have been enhanced as well. In

Figure 28, PCA has dropped out the few static false targets from Figure 27 but does not provide any fidelity on target detections for 3 or 5 dB SNR. The best method, by far, is to enhance the image with LCM and then run PCA. Figure 29 shows very clear lines of detection with much fewer false detects for all four SNR values.

Figures 29 to 31 illustrate the DR, PR, and number of false alarms calculated for the three simulations above for scenario one.

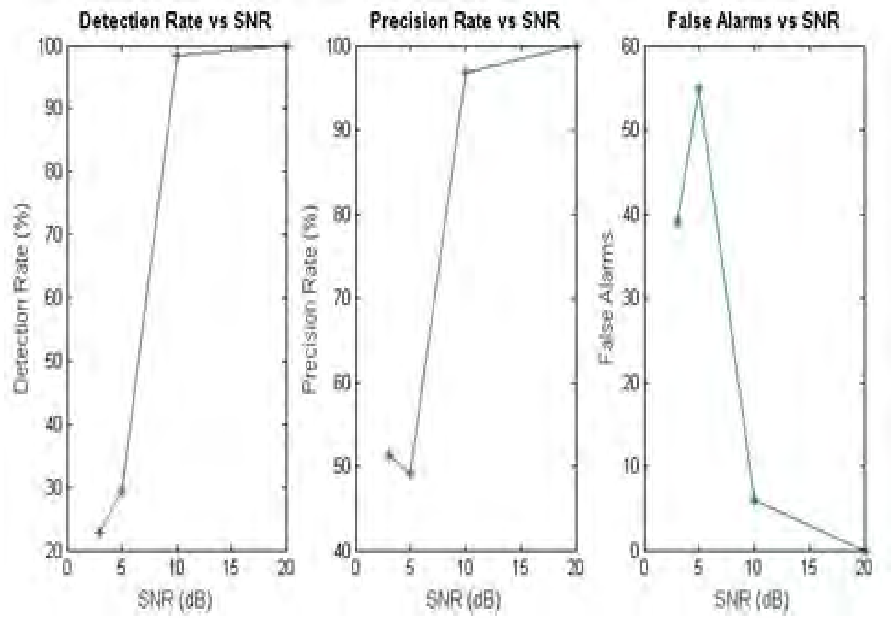

Figure 29. LCM detection metrics for scenario one.

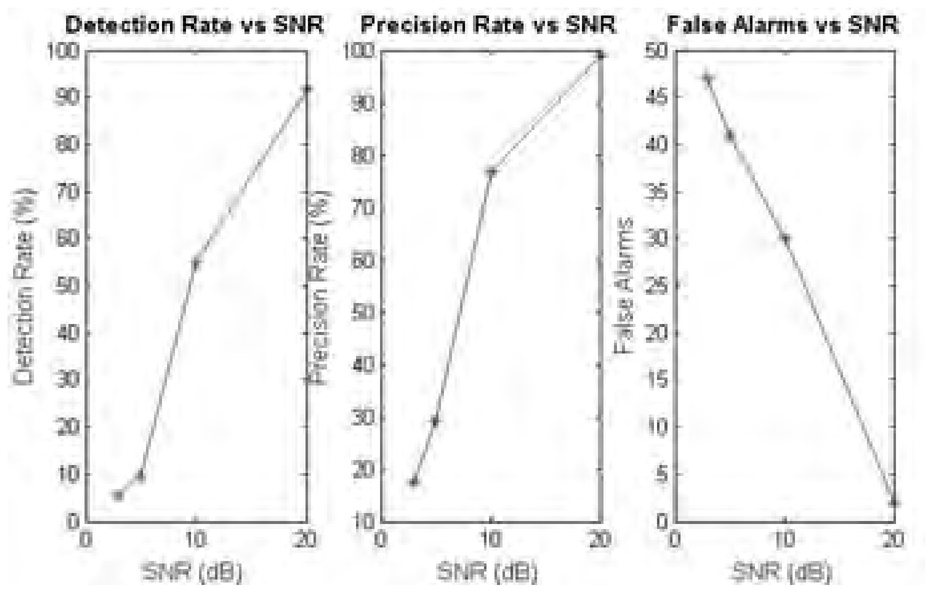

Figure 30. PCA detection metrics for scenario one

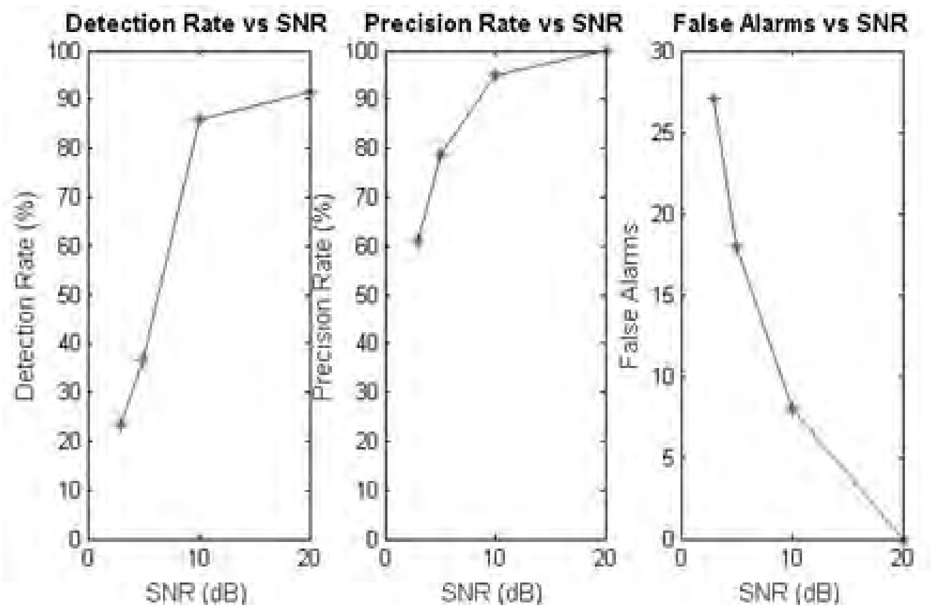

Figure 31. LCM and PCA metrics for scenario one

The above figures clearly indicate that a combination of the two pre-detection methods is preferred. DR varies slightly but still rises dramatically with SNR in all three simulations. The clear advantage is in PR and number of false alarms. The third pre-detection simulation still had a 60 percent PR at its lowest SNR value. In addition, it registers only 27 false alarms whereas LCM and PCA indicate 40 and 48 respectively.

The same three simulation events above will now be tested against scenario two. Expected results should be similar with slight break down in performance when target tracks intersect each other.

2. Multiple Crossing Targets

Scenario two involves the same three large ships as targets, but now their paths cross. This will provide detail on how the pre-detection and tracking algorithms perform when targets are close together.

Figures 32 to 34 show true trajectories and target detections using LCM, PCA, and LCM then PCA respectively for each of the four SNR values being compared.

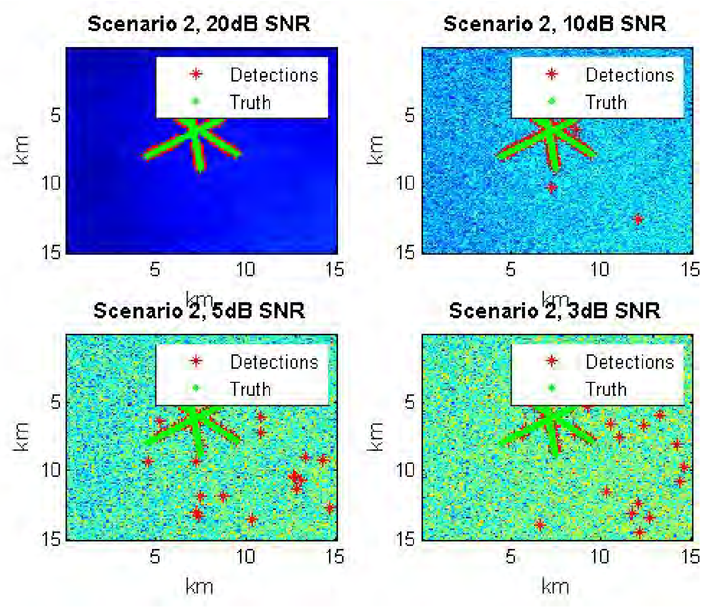

Figure 32. Local contrast method applied to scenario two.

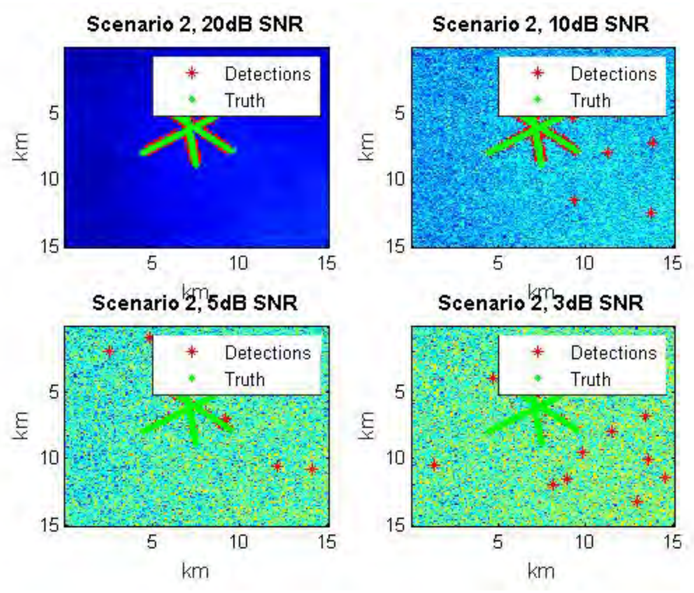

Figure 33. Principal component analysis applied to scenario two.

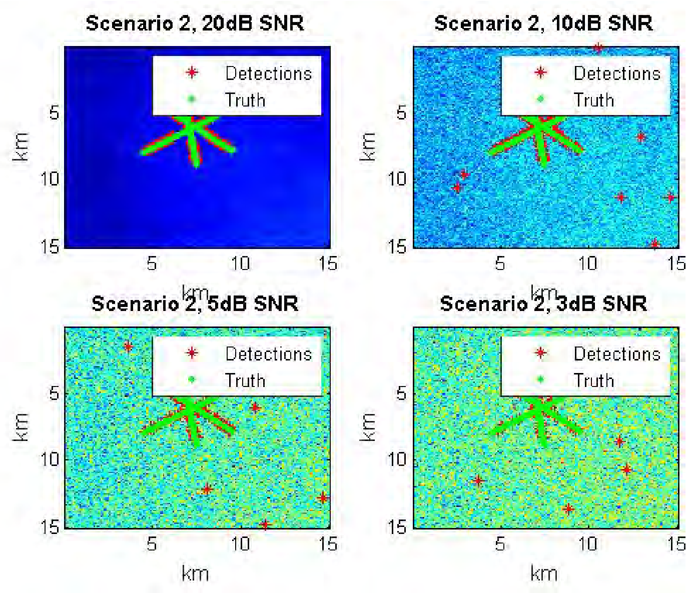

Figure 34. Combination pre-detection algorithm applied to scenario two.

Scenario two detection results reflect almost the same findings as scenario one. All three simulations get progressively worse as the noise level increases. Using LCM followed by PCA again provides the best detection resolution, but at the 3 dB SNR even this method begins to break down as the three targets intersect each other. At this point, the algorithm has a difficult time distinguishing between the three and effectively reports only one or two detects.

Figures 35 to 37 illustrate the DR, PR, and number of false alarms calculated for the three simulations above for scenario two.

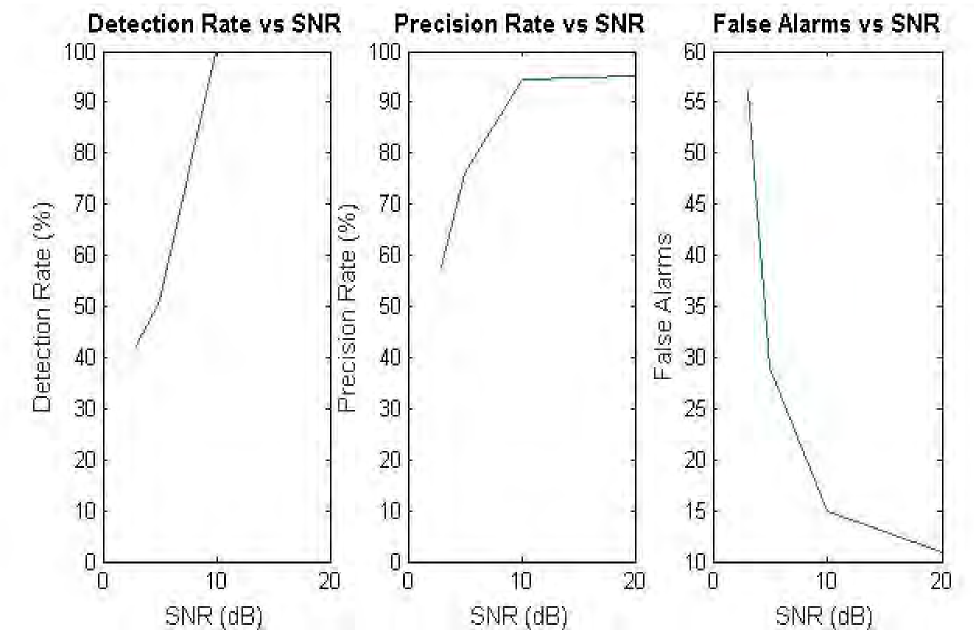

Figure 35. LCM detection metrics for scenario two.

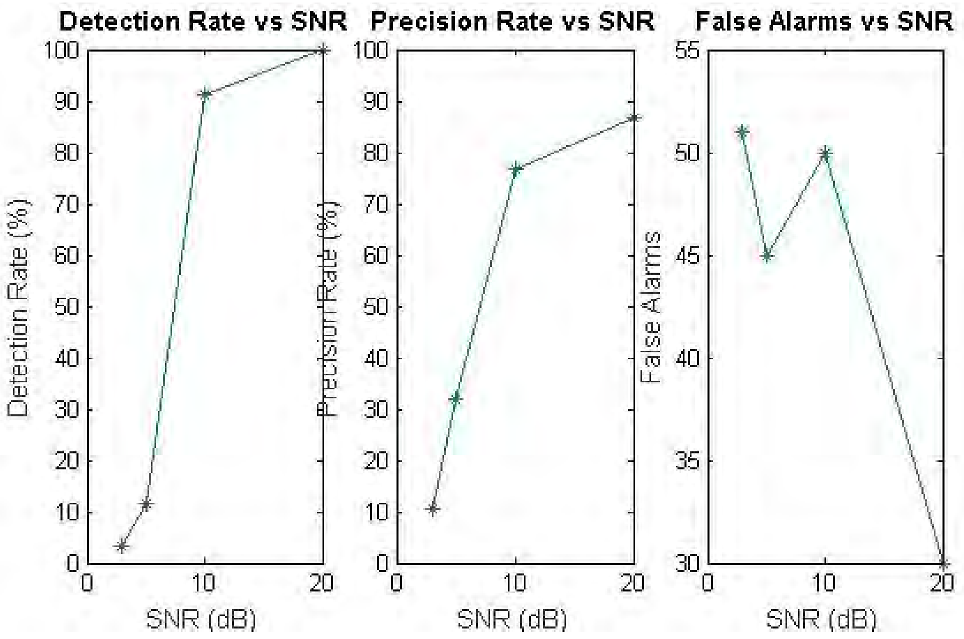

Figure 36. PCA detection metrics for scenario two.

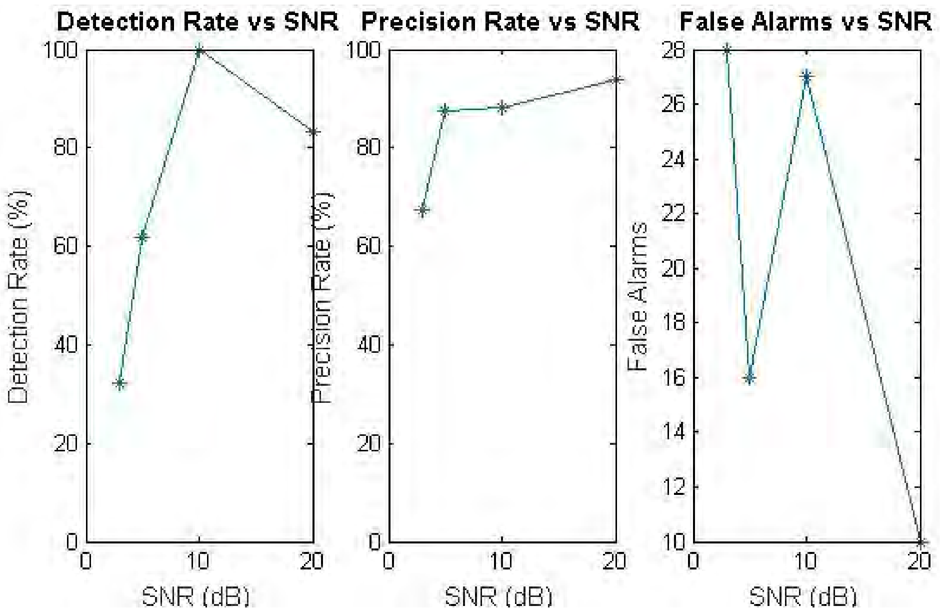

Figure 37. LCM and PCA metrics for scenario two.

Once again, metrics behave favorably for this scenario. Detection rate is slightly lower than in scenario one due to the lack of target resolution near the end of the simulation when the tracks intersect. This causes a drop in precision as well. False alarm values are approximately equal to the first scenario with a max of 28 at 3dB and none at 20dB.

3. Swarm Attack against HVU

This culminating scenario is one that the Navy worries about every day. There is a HVU, most likely a carrier or amphibious ship, transiting the SOH with constant velocity and heading. Multiple attacking targets all pursue from varying locations to swarm the HVU and intercept all at the same time. This simulation will involve eight attackers whose trajectories require changes in velocity and acceleration in order to all reach the HVU at the same time. This is a scenario in which detection rate and resolution in detection system play a big role. Inevitably, the detection algorithm will begin grouping targets together and no longer be able to distinguish all eight targets at some point. The goal is to be able to hold out until the targets are at least within visible range of a HVU. Each simulation will include both Poisson noise and white Gaussian noise. Noise is varied to generate desired SNR.

Figures 38 to 40 illustrate the detections versus ground truth for the swarm scenario. The detections for both the HVU and attackers are in red while the true trajectories are in green.

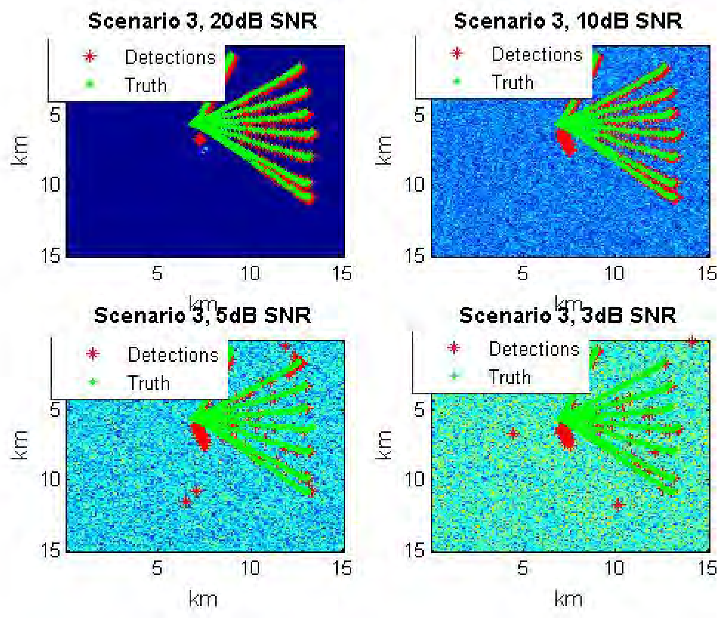

Figure 38. Local contrast method applied to scenario three.

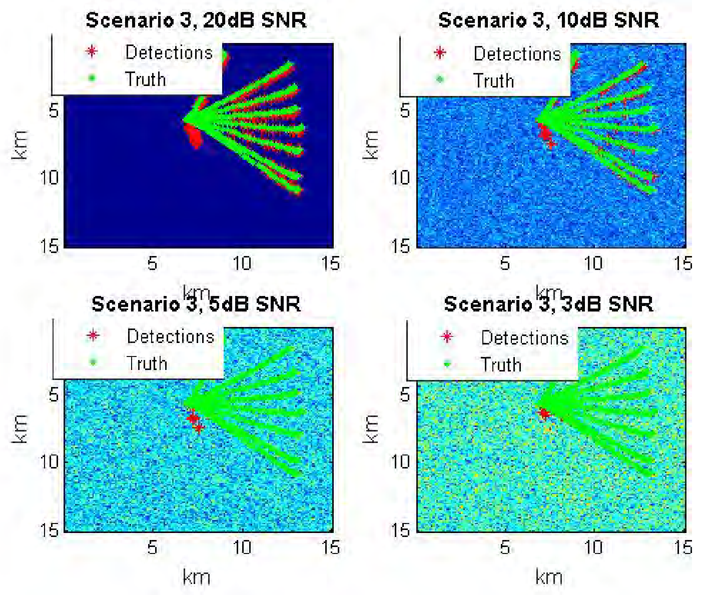

Figure 39. Principal component analysis applied to scenario three.

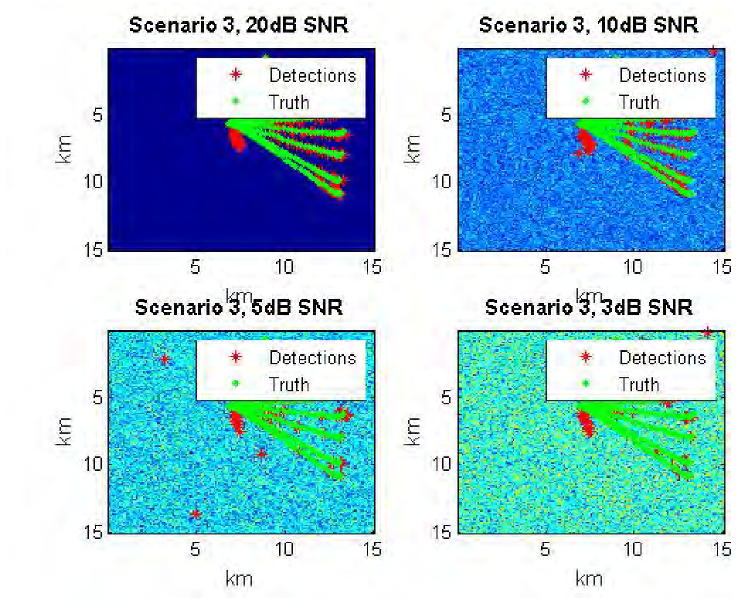

Figure 40. LCM and PCA applied to scenario three.

The swarm scenario does appear to present a significant challenge to both LCM and PCA at lower SNR values with almost no detections for PCA lower than 10dB. While the third simulation using both algorithms does clean up the image slightly as before, the higher speeds of the targets and their converging trajectories result in very few true detections. A closer examination of detection metrics further supports these findings.

Figures 41 to 43 show how detection, precision, and false alarms were compared to SNR in each of the three simulations for scenario three.

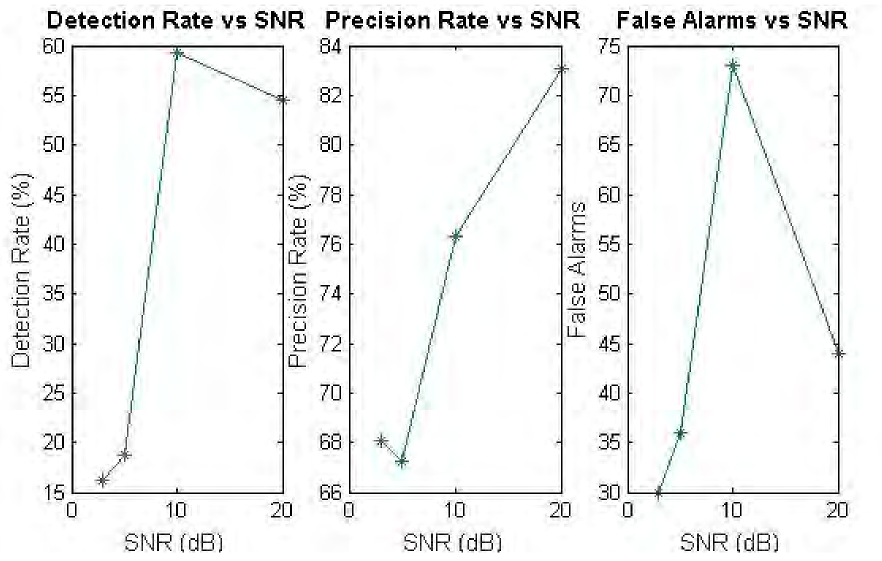

Figure 41. LCM detection for scenario three.

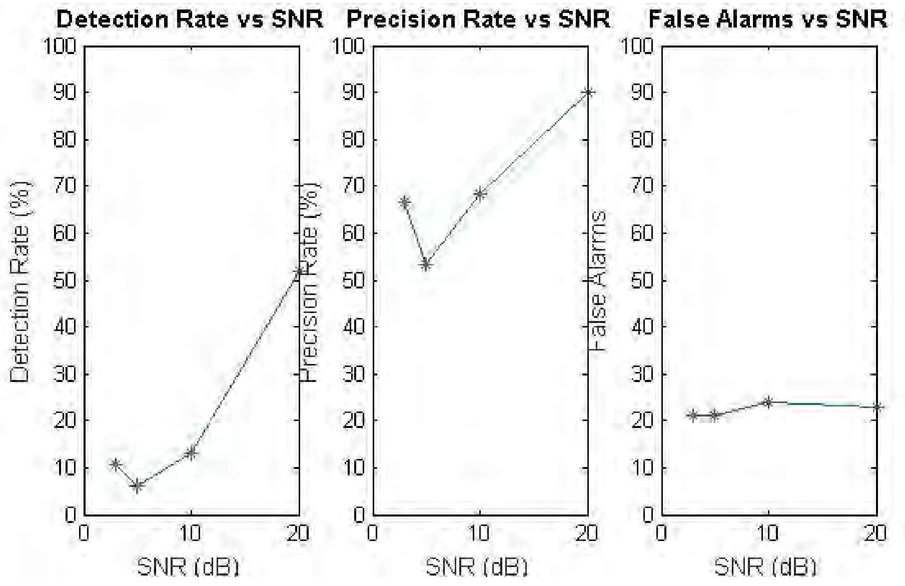

Figure 42. PCA detection for scenario three.

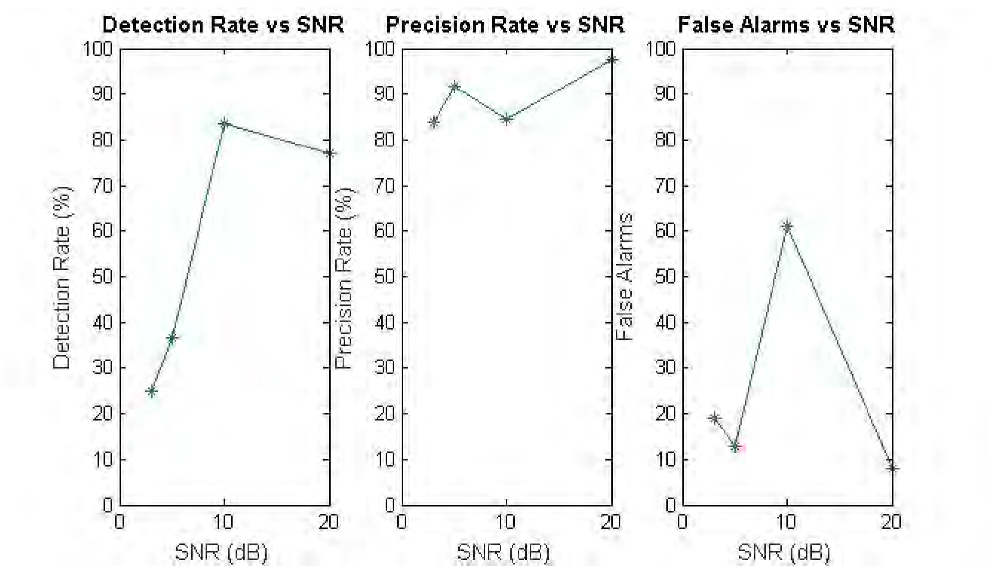

Figure 43. LCM and PCA applied to scenario three.

Using just the LCM or PCA method for this swarm scenario would not produce valuable data. Maximum detection rates are 60 and 50 percent, respectively, with precision only getting just above 80 percent. In addition, the LCM produces almost 75 false alarms at 10dB and nearly 45 at 20dB. This performance would not be suitable to provide valuable intelligence on fast moving targets. The third simulation again gives superior performance to the other two, but still has problems. Detection rates are still low even at 20dB SNR which would be highly unlikely in a real-world scenario. Precision is much higher and false alarms are more reasonable.

B. TRACKING

Tracking performance for each of the three scenarios is also researched at a high level. Simple centroid tracking with a Kalman filter is implemented. The filter is initialized from the first detection, plant noise is zero, and a gate size of 13 pixels is chosen. The simulations compared use LCM and PCA as pre-detection methods in order to provide a consistent best-case detection rate to feed to the Kalman filter. This also allows comparison of just the tracking algorithm. Figures 44 to 46 show how the simple tracking algorithm performed in each of the three scenarios for SNR values of 20, 10, 5, and 3dB. The solid lines are the Kalman predicted tracks and the stars are the measurements of detections given by the pre-detection algorithm.

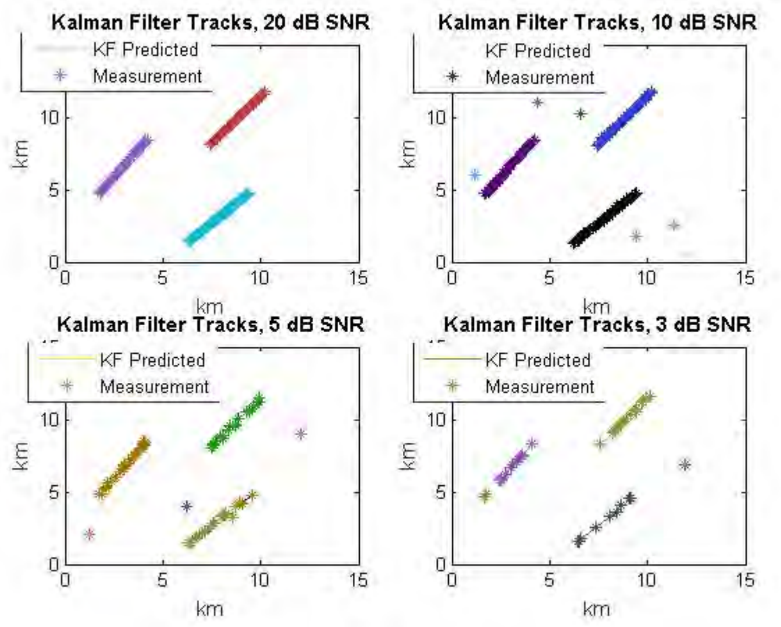

Figure 44. Tracking results for scenario one.

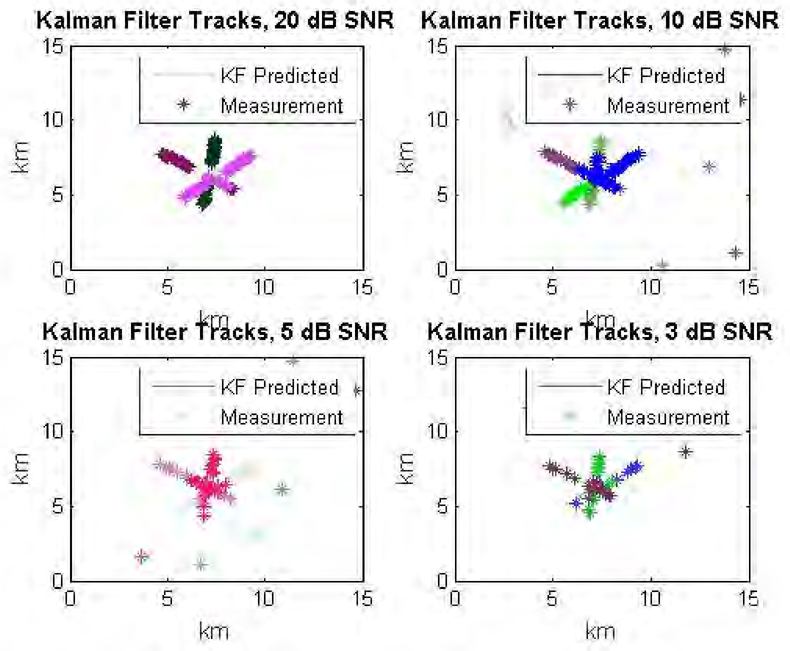

Figure 45. Tracking results for scenario two.

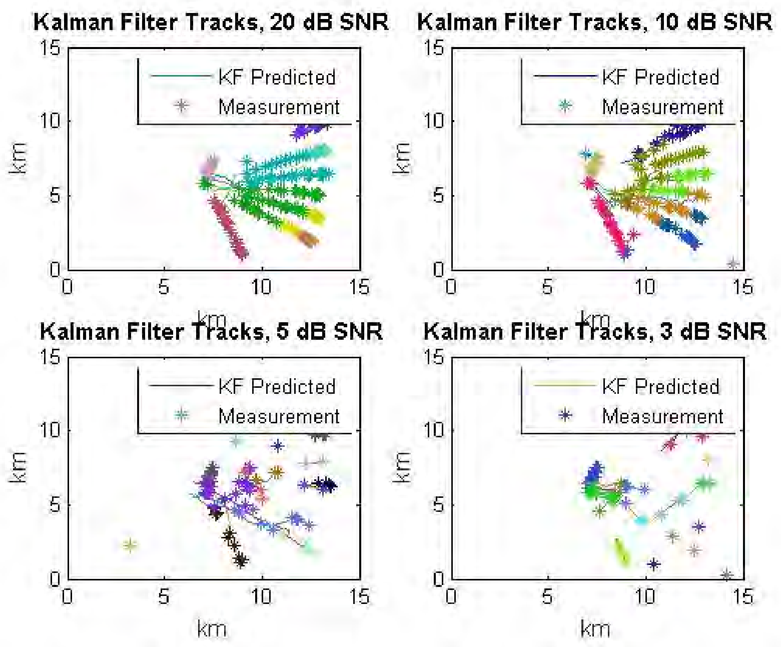

Figure 46. Tracking results for scenario three.

The Kalman filter tracker performs extremely well on scenarios one and two. In both these cases the targets are slow and moving at a constant velocity and heading. This is why even when the tracks intersect in scenario two the tracker has enough *a priori* knowledge to continue each track straight without trying to associate it to one of the other tracks. This is also true for scenario three at the two higher SNR values. The filter continues the majority of the tracks into the converging point even after the separate detections disappear. Finally, at the lowest SNR values detections are random and scattered creating problems in initiating correct tracks. When the ships converge to create a giant target, the detections increase but no track knowledge exists. This phenomenon is most likely the result of too few detections and data association errors. The significant takeaway is that a centroid tracker with a Kalman filter still provides enough fidelity even in a swarm scenario to provide valuable information. The attackers are within two kilometers of the HVU before the algorithm breaks down in most cases. This is close enough where the HVU will either have the targets in visual range or can begin to track them with their organic radar assets.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

A. SUMMARY

This thesis has presented a set of detection and tracking methods suitable for tracking of multiple moving ground targets from space-based sensors or ground-based high-energy lasers. The local contrast method and principal component analysis pre-detection algorithms have been compared and both have advantages and disadvantages when working in a low SNR environment. LCM is the better choice as a standalone because of the signal enhancement capability which raises the SNR of your target. There is more opportunity for detection. The significant drawback is that large static objects will be seen as false detects. PCA works very well for subtracting out static objects and performing detection only on a foreground of moving pixels. However, in the principal component breakdown the maximum intensityis lost and your SNR effectively decreases. This results in much fewer detects at any SNR than LCM and poor performance below 5 dB. The most advantageous method examined was to enhance the signal and suppress clutter with LCM and then run that image through the PCA algorithm. This method saw a significant increase in detection rate, precision rate, and a drop in false alarms. This method also had a significant impact on tracking performance.

Two of the three common tracking methods were researched. Centroid and correlation both have advantages and disadvantages and the method of choice depends on the application. The standard Kalman filter and alpha-beta filter both keep the root mean square error in track prediction above 70 percent on constant velocity and/or constant acceleration linear models. They, too, have difficulty once the problem becomes nonlinear. The Kalman filter used in the three scenarios tested had great results when the combinations of the two pre-detection methods were used. This gave enough true detects to the tracker to estimate the correct track even after the detections deteriorated when targets converged. Precise tracks were identified down to 3 dB for all scenarios with minor expected breaks in the track for scenario three when all attackers converged on a single point. This presents data association problems for the tracking algorithm. Advanced techniques can help with this issue.

The standard routines and filters above work favorably against the relatively benign scenarios presented. The data association problem presents a significant challenge in a cluttered environment like the SOH or in a scenario where multiple targets get close to each other. When multiple detections fall within the same error ellipse of the tracker or within the same gate, advanced techniques may be needed to solve this issue. The extreme solution is to run a full multi-hypothesis tracker (MHT) in which every detection is propagated separately as if it is associated with every track. This continues every frame until sometime in the future where those propagations become unrealistic. This problem can easily expand exponentially and create an algorithm too computationally expensive to implement [19].

Constraints can be added to the MHT problem to come up with a more reasonable solution. Examples are roads or shipping lanes when tracking ground targets over land or sea. It is highly likely that a car is traveling on a road and in crowded areas like the SOH most large ships stick to the designated shipping lanes. These roads or shipping lanes can be overlaid onto the image to provide boundaries in addition to gates. It can significantly prune the MHT problem by dismissing a hypothesis once it is propagated outside that constraint [20]. The other highly effective technique is to use a VMF when tracking relatively constant velocity targets like cars or ships. The speed of a car on a road or ship over water can be estimated pretty accurately. The VMF applies this constraint over multiple frames to further constrain the MHT problem. This significantly aids in tracking, but can also provide a boost in detecting low SNR targets since it is integrating over multiple frames and not just one image [21].

Finally, once constraints have been put in place a method is needed to analyze the "cost" of assigning a hit to a certain track. Then, these costs can all be run through Munkres' version of the Hungarian algorithm to determine the optimal cost value to each hypothesis. This will, in theory, associate the detections to the tracks which require the least deviation from the model. This, of course, could break down if cars or ships are performing erratic maneuvers. This algorithm is another optimal solution to pruning the MHT issue [22].

B. FUTURE WORK

This work should provide a good starting point for future students to conduct research on target detection and tracking within the SRDC AOCoE. The next step for detection research is to look into more complex change detection methods. Sandia National Laboratories is also conducting research and has done some work using different spatial and temporal models to accomplish this. Students will also have the opportunity in the near future to work with Boeing in developing realistic tracking algorithms that can be implemented on actual hardware via field-programmable gate arrays (FPGAs). These tracking algorithms can then be tested first in the laboratory and then in the field on the HBCRT once it arrives. There is also considerable work to be done in creating more robust tracking algorithms that can better handle some of the issues presented here. There are several government experts currently working on MHT, VMFs, and geographical constraints. These experts have demonstrated success in overlaying these geographical constraints onto the MHT problem.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. MATLAB CODE

```
% Scene_Generation
% 12/3/14
%
% Scene  - Strait of Hormuz
% Landsat SWIR Data
% GSD is 30 meter / pixel
%
% Image resized to 100 meter/pixel
% Band 5 - Near Infrared (NIR)  0.85 - 0.88 30m
% Band 6 - SWIR 1   1.57 - 1.65 30m
% Band 7 - SWIR 2   2.11 - 2.29 30m
% Band 8 - Panchromatic 0.50 - 0.68 15m
%
% The script limits the top speed of the ship to 32 knots
%
% Script prompts user to pick a region of the image to process
% Script prompts user to pick starting and ending points of track
%
% Output:
%
%   Scenario_1  - [pixels, samples, SNR Simulations ],
%   position_true - [sampes x 2 x number of ships]
%
% --------------------------------------------------------------------
--

close all
clear all


[A, R] = geotiffread('LC81590422014115LGN00_B6.TIF'); % Load SWIR Band
5
A = imresize(A,.333);    % Resize to 100m/pixel
A = imrotate(A,12.6);    % Rotate Image
A = double(A(461:2460,472:2471)); % Turn to double

load('ship.mat')
frame_rate = .2; % frames/second

figure(1)
imagesc(A);


disp('Select top left corner of image chip:')
[y(1), x(1)] = ginput(1);
prompt = 'Choose length size of image chip (<150 pixels): ';
chip_size = input(prompt);
if chip_size > 150
    chip_size = 150;
end
```

```
y(2) = y(1)+chip_size;
x(2) = x(1)+chip_size;
x=round(x);
y=round(y);

A = A(min(x):max(x)-1,min(y):max(y)-1);

figure(1)
imagesc(A)

prompt = 'How many tracks would you like to generate? ';
track_number = input(prompt);

prompt = 'How long is the simulation (seconds)? ';
sim_time = input(prompt);

% prompt = 'How many samples would you like to generate? ';
% nsamples= input(prompt);

nsamples = floor(sim_time*frame_rate);
disp(['Samples being generated:', num2str(nsamples)]);

disp('Click on the start locations of the track(s)');
[xt ,yt] = ginput(track_number);

disp('Click on the end locations of the track(s)');
[xt2, yt2] = ginput(track_number);


% Compute Heading and speed
hdg = atan2(xt2-xt,yt2-yt);     % Radians
speed = (sqrt((yt2-yt).^2+(xt2-xt).^2))/(sim_time); % pixels/second
max_speed = 32*.5144*(1/100); % Max speed of 32 knots converted to
pixels/sec
% Limit max speed to 32 knots
speed_index = find(abs(speed>max_speed));
speed(speed_index)=max_speed;
%sigma_x = .1;
%sigma_y = .1;
%sigma=[sigma_x;sigma_y];
delta = sim_time/nsamples;

%% Generate Tracks
% Define Transition Matrix - F

F=[1,delta,0,0;
    0,1,0,0;
    0,0,1,delta;
    0,0,0,1];

% Define Measurement Matrix - H
H=[1,0,0,0;0,0,1,0];
```

```
% Initialize outputs and error matrices
zout=[];             % measurement output matrix
posout=[];           % true position output matrix
position_true=[];    % truth history (nx2xships)


for i = 1:track_number
   xi=[xt(i);speed(i)*sin(hdg(i));yt(i);speed(i)*cos(hdg(i))];
%Initial state
   x=xi;               % set state to initial states

   for j = 1:nsamples
   ztrue=H*x;
   posout=[posout,ztrue];

   % Add Noise
   %z=ztrue+randn(size(sigma)).*sigma;
   z = ztrue;

   % Add the mesurement to the measurement output matrix
   zout=[zout,z];

   % Target motion step
   x=F*x;

   end
    x_track(:,i) = round(zout(1,:))';
    y_track(:,i) = round(zout(2,:))';

    zout=[];             % measurement output matrix
    position_true(:,:,i) = posout';
    posout=[];
end

%% Build Scene

    % SNR is defined as 20log10(mu_target/sigma_background)
    % mu is the average power of the target
    % sigma is the standard deviation of the background

    A = mat2gray(A);  % Convert to gray
    A_org = A;
    mu = mean2(ship(2:4,1:4));
    var_A = var(A(:));
    % Set SNR to be 20dB based on initial image
    mu_20dB = 10*std2(A);
    k_ship = mu_20dB/mu;
    ship = ship.*k_ship;
    mu = mean2(ship(2:4,1:4));
    SNR_Initial = 20*log10(mu/std2(A))

    % Determine Variance to add to white Noise
    var1 = 0;                     % 20 dB
    var2 = (mu/10^.5)^2;         % 10 dB
    var3 = (mu/10^(4/20))^2;     % 5 dB
    var4 = (mu/10^(3/20))^2;     % 3 dB

for p = 1:4         % Run Different Noise Scenarios

  switch p
      case 1
          variance = var1;
      case 2
          variance = var2-var_A;
      case 3
          variance = var3-var_A;
      case 4
          variance = var4-var_A;
  end


% Add noise to image - These Can be Commented Out
%A = imnoise(A,'poisson');
%A = imnoise(A,'gaussian',0,var);

    for i = 1:nsamples

%Insert Ships Into Scene
        %A = imnoise(A,'gaussian',0,variance);
        seed = randn(size(A));
        del_img = sqrt(variance)*seed;
        A = A + del_img;

        %var_A = var(A(:));
        SNR = 20*log10(mu/std2(A))
        for j = 1:track_number

        A(y_track(i,j)-2:y_track(i,j)+2, x_track(i,j)-
2:x_track(i,j)+2)=A(y_track(i,j)-2:y_track(i,j)+2, x_track(i,j)-
2:x_track(i,j)+2)+ship;

        end
    Scenario_1(:,i,p)=mat2gray(A(:));

 imagesc(A)
 pause(.1)
 A = A_org;

    end

end
% Scenario_1_gold= Scenario_1;
% position_true_gold = position_true;
%
% save Scenario_1_gold.mat Scenario_1_gold;
% save position_truth_gold.mat position_true_gold;
```

```
save Scenario_1.mat Scenario_1;
save position_truth.mat position_true;




% Scene_Generation
% 11/22/14
%
% Scene  - Strait of Hormuz
% Landsat SWIR Data
% GSD is 30 meter / pixel
%
% Image resized to 100 meter/pixel
% Band 5 - Near Infrared (NIR)  0.85 - 0.88 30m
% Band 6 - SWIR 1   1.57 - 1.65 30m
% Band 7 - SWIR 2   2.11 - 2.29 30m
% Band 8 - Panchromatic 0.50 - 0.68 15m
%
% This script uses a Directmethods appraoch to generate a swarm attack
% for multiple attackers starting a different starting locations. The
trajectories
% assume that all attackers arrive at the same location at the same
final time.
%
% p0 = initial position of attacker
% p0ship = initial position of HVU
% PsiShip = heading of HVU
% v0 = initial velocity of attacker
% vf = final velocity of attacker
% vmax = max velocity of attacker
% vmin = min velocity of attacker
% psidotmax = max turn rate (radians/time)
%
% function / script calls: Attack_trajectory.m
%                          DirectMethodAttackPlot.m
%
% The script calls the detection algorithm PCA_Detect.m to perform the
% principle component analysis. This function returns the foreground
% images.  The function requires you to input the scenes in vector
format
% as well as the window size, w.  You can also set the variable LCM to
1 to
% turn on the local contrast method algorithm.  The LCM algorithm can
be
% used to improve the contrast of the forground targets to aid in
% Detection.
%
% The function Target_Detection.m is used to detect targets from the
% foreground images.  Inputs include the foreground images, a threshold
gain
% typically between 1 and 5, and maximum and minimum target lenghths to
% filter returns by size (typically between 2 and 6).
%
% You can also add LCM as the general pre-detection algorithm by
% uncommenting LCM_Enhance Function
%
% Output:
% Target_Detects - Cell Array of Possible targets for each image
%                  Use {} to access cell array
%     Example:  Target_frame1 = Target_Detects{1}
% ---------------------------------------------------------------------
--

close all
clear all
clc

% Initialize and Set Constants

Process_LCM = 1;                % Set to 1 to turn on
Process_PCA = 0;                % Set to 1 to turn on
gate_size = 5;                  % Gate size to associate detect with KF
prediction
k_thresh = 3;                          % Detection gain
PCA_then_LCM = 0;               % Run LCM after running PCA detection
min_bb = 3;                     % Minimum bounding box size
max_bb = 12;                    % Maximum bounding box size
frame_rate = .2;                % Sensor frame rate (Hz)
chip_size = 150;                % Chip_size
lb_window = 6;                  % PCA look back window
% SNR = 5;                       % Set SNR

% define initial and final conditions, limits on heading and velocity
global v0 vf vmin vmax psidotmax p0 p0ship psiShip vShip a0 af amin
amax N



% Initialize Image
[A, R] = geotiffread('LC81590422014115LGN00_B6.TIF'); % Load SWIR Band
5
A = imresize(A,.333);    % Resize to 100m/pixel
A = imrotate(A,12.6);    % Rotate Image
A = double(A(461:2460,472:2471)); % Turn to double
A = A(401:1400, 201:1200);
A = mat2gray(A);  % Convert to gray
[Ax,Ay]=size(A);

%figure(1)
%imagesc(A)

% Define HVU image
HVU = A(65:72,151:156);
HVU = imrotate(HVU,105);
HVU = HVU(2:8, 4:10);

% Define Attackers Image
```

```
load('ship.mat')


%Initial Position of Attackers
prompt = 'How attackers would you like to generate? ';
number_of_attackers = input(prompt);

p0ship = [600;500];  % location of HVU
psiShip = -3/5*pi;  % Heading of HVU

figure(1)
imagesc(A(p0ship(2)-74:p0ship(2)+75, p0ship(1)-74:p0ship(1)+75));
A_chip = (A(p0ship(2)-74:p0ship(2)+75, p0ship(1)-74:p0ship(1)+75));

disp('Click on the start locations of the track(s)');
p0 = ginput(number_of_attackers);

p0=p0';
p0 = [p0(1,:)-74+p0ship(1); p0(2,:)-74+p0ship(2) ];

v0 = 10*.5144/100;   % Convert to pixels/sec assuming 100 m / pixel
vf = 15*.5144/100;
a0 = 0;
af = 0;
vShip = 20*.5144/100;
vmax = 30*.5144/100;
vmin = 5*.5144/100;
amax = .147*4;
amin = 0;
psidotmax = .2;

% these are max and min values computed by directmethods
global psidot_max v_min v_max a_min a_max
data=[];

[x,feval] = fminsearch('Attack_trajectory',[300]);
tf=x;

c1=(v0-vf)/tf;
c2=v0;

    if v0 == vf
         tauf = c2*tf;
    else
         tauf=(c2*exp(c1*tf)-c2)/c1;
    end

% Chose number of iterations
%N=round(tf);
N = floor(tf*frame_rate);  % Choos

DirectMethodsAttackPlot
```

```
disp(['Simulation time = ', num2str(tf)]);

% Plot Scene
figure(2)
imagesc(A)
hold on
plot(HVU_track(:,1),HVU_track(:,2),'g')
hold on

for ii = 1:number_of_attackers

    plot(Attacker_tracks(:,1,ii),Attacker_tracks(:,2,ii))
    hold on

end


%% Build Scene

A_org = A;

% Format track data add variance, variance not included here

%sigma_x = .1;
%sigma_y = .1;
%sigma=[sigma_x;sigma_y];
x_track = zeros(N+1, number_of_attackers);    %Initialize Matrix
y_track = zeros(N+1, number_of_attackers);    %Initialize Matrix

Attacker_true = Attacker_tracks;
HVU_true = [HVU_track(:,1), HVU_track(:,2)];                 % true
position
HVU_x = round(HVU_true(:,1))';
HVU_y = round(HVU_true(:,2))';


for i = 1:number_of_attackers
       y_track(:,i) = round(Attacker_tracks(:,2,i));             %
Round to pixel
       x_track(:,i) = round(Attacker_tracks(:,1,i));
end

for c = 1:4         % Run Different Noise Scenarios

    switch c
        case 1
            SNR=20;
        case 2
            SNR=10;
        case 3
            SNR=5;
        case 4
            SNR=3;
    end

% Introduce Noise
    A_chip = mat2gray(A_chip);  % Convert to gray
    A_chip_org = A_chip;
    mu = mean2(ship(2:4,1:4));
    var_A = var(A_chip(:));
    mu_20dB = 10*std2(A_chip);
    k_ship = mu_20dB/mu;
    ship = ship.*k_ship;
    HVU = HVU.*k_ship;
    mu = mean2(ship(2:4,1:4));
    SNR_Initial = 20*log10(mu/std2(A_chip))
    var2 = (mu/10^(SNR/20))^2;
    variance = var2-var_A;


    if SNR_Initial <= SNR
        SNR = SNR_Initial;
        variance = 0;
    end

 for i = 1:N       % Number of iterations

        A = A_org;
        % Add noise to image

        seed = randn(size(A));
        del_img = sqrt(variance)*seed;
        A = A + del_img;
        SNR_current = 20*log10(mu/std2(A(p0ship(2)-74:p0ship(2)+75,
p0ship(1)-74:p0ship(1)+75)))

        A(HVU_y(i)-3:HVU_y(i)+3, HVU_x(i)-3:HVU_x(i)+3)=A(HVU_y(i)-
3:HVU_y(i)+3, HVU_x(i)-3:HVU_x(i)+3)+HVU;
    for j = 1:number_of_attackers

        A(y_track(i,j)-2:y_track(i,j)+2, x_track(i,j)-
2:x_track(i,j)+2)=A(y_track(i,j)-2:y_track(i,j)+2, x_track(i,j)-
2:x_track(i,j)+2)+ship;

    end

chip5 = A(p0ship(2)-74:p0ship(2)+75, p0ship(1)-74:p0ship(1)+75);

chip_hist(:,i,c) = mat2gray(chip5(:));

figure(3)
imagesc(chip5)
pause(.1)

 end
end
```

```
  if Process_LCM == 1

% -------- LCM Pre-Detection Begin --------------
[chip_hist] =  LCM_enhance(chip_hist);
Target_Detects = Target_Detection(chip_hist,k_thresh,min_bb,max_bb);
figure(4)
for i = 1:size(chip_hist,2)
imagesc(reshape(chip_hist(:,i),chip_size,chip_size))
hold on
    frame_targets = Target_Detects{i};
    num_targets = size(frame_targets,1);
    for p = 1:num_targets
        BB = frame_targets(p,:);
    rectangle('Position',
[BB(1),BB(2),BB(3),BB(4)],'EdgeColor','g','LineWidth',1 )
    end
pause(.1)
end
%-------- LCM Pre-Detection End --------------
 end

if Process_PCA == 1

%-------- PCA Pre-Detection Begin --------------
[Fg_chip_hist] = PCA_Detect(chip_hist,lb_window,PCA_then_LCM); % Set
last value to 1 to turn on LCM after PCA
Target_Detects = Target_Detection(Fg_chip_hist,k_thresh,min_bb,max_bb);

% Plot Foreground and Detections, and compute centroid locations

figure(4)
for i = 1:size(Fg_chip_hist,2)
imagesc(reshape(Fg_chip_hist(:,i),chip_size,chip_size))
hold on
    frame_targets = Target_Detects{i};
    num_targets = size(frame_targets,1);
    for p = 1:num_targets
        BB = frame_targets(p,:);
    rectangle('Position',
[BB(1),BB(2),BB(3),BB(4)],'EdgeColor','g','LineWidth',1 )
     end
pause(.1)
end

%-------- PCA Pre-Detection End --------------

end

%---------- Tracking -------------------------
```

```
Td_c = c_detects(Target_Detects);

dd = Td_c;
save dd.mat dd;     % Save detections as centriods
Test_KF2




%%DT_Analysis

% Simulations
close all
clear all
load('Scenario_1_gold.mat')
load('position_truth_gold.mat')
Scenario_1 = Scenario_1_gold;
position_true = position_true_gold;

n_scenarios = size(Scenario_1,1);

for snr_scenario = 1:4

Scenario = snr_scenario;                             % Set Scenario to
process
Process_LCM = 0;                        % Set to 1 to turn on
Process_PCA = 1;                        % Set to 1 to turn on
gate_size = 13;                         % Gate size to associate detect
with KF prediction
k_thresh = 3;                           % Detection gain
PCA_then_LCM = 0;                       % Run LCM after running PCA
detection
min_bb = 3;                             % Minimum bounding box size
max_bb = 12;                            % Maximum bounding box size
chip_size = sqrt(size(Scenario_1,1));   % Chip_size
lb_window = 6;                          % PCA look back window

chip_hist = Scenario_1(:,:,Scenario);

if Process_LCM == 1

% -------- LCM Pre-Detection Begin --------------
[chip_hist] =  LCM_enhance(chip_hist);
Target_Detects = Target_Detection(chip_hist,k_thresh,min_bb,max_bb);
figure(4)
for i = 1:size(chip_hist,2)
imagesc(reshape(chip_hist(:,i),chip_size,chip_size))
hold on
    frame_targets = Target_Detects{i};
    num_targets = size(frame_targets,1);
    for p = 1:num_targets
        BB = frame_targets(p,:);
    rectangle('Position',
[BB(1),BB(2),BB(3),BB(4)],'EdgeColor','g','LineWidth',1 )
```

```
    end
%pause(.1)
end
%-------- LCM Pre-Detection End --------------
end

if Process_PCA == 1

%-------- PCA Pre-Detection Begin --------------
[Fg_chip_hist] = PCA_Detect(chip_hist,lb_window,PCA_then_LCM); % Set
last value to 1 to turn on LCM after PCA
Target_Detects = Target_Detection(Fg_chip_hist,k_thresh,min_bb,max_bb);

% Plot Foreground and Detections, and compute centroid locations

figure(4)
for i = 1:size(Fg_chip_hist,2)
imagesc(reshape(Fg_chip_hist(:,i),chip_size,chip_size))
hold on
    frame_targets = Target_Detects{i};
    num_targets = size(frame_targets,1);
    for p = 1:num_targets
        BB = frame_targets(p,:);
    rectangle('Position',
[BB(1),BB(2),BB(3),BB(4)],'EdgeColor','g','LineWidth',1 )
     end
%pause(.1)
end

%-------- PCA Pre-Detection End --------------

end

%---------- Tracking -------------------------

Td_c = c_detects(Target_Detects);

dd = Td_c;
save dd.mat dd;     % Save detections as centriods
Test_KF2

%---------------- Detection Rate -----------------

for i = 1:size(tracks,2)
Detect_hist(i) = tracks(i).age;
end
Total_Detects = sum(Detect_hist)
True_Detects = sort(Detect_hist);
True_Detects = sum(True_Detects(end-2:end))

DR(snr_scenario) = True_Detects/180;
PR(snr_scenario) = True_Detects/Total_Detects;
FA(snr_scenario) = Total_Detects-True_Detects;
```

```
%---------------- PLOTS-----------------



end

%Plot SNR Target and Background

figure(1)
subplot(2,2,1)
resol = .1; % 1 km
Im1 = reshape(Scenario_1(:,1,1),chip_size,chip_size);
imagesc([resol resol*chip_size],[resol resol*chip_size],Im1)
title('Scenario 1, 20dB SNR')
xlabel('km')
ylabel('km')
subplot(2,2,2)
resol = .1; % 1 km
Im1 = reshape(Scenario_1(:,1,2),chip_size,chip_size);
imagesc([resol resol*chip_size],[resol resol*chip_size],Im1)
title('Scenario 1, 10dB SNR')
xlabel('km')
ylabel('km')
subplot(2,2,3)
resol = .1; % 1 km
Im1 = reshape(Scenario_1(:,1,3),chip_size,chip_size);
imagesc([resol resol*chip_size],[resol resol*chip_size],Im1)
title('Scenario 1, 5dB SNR')
xlabel('km')
ylabel('km')
subplot(2,2,4)
resol = .1; % 1 km
Im1 = reshape(Scenario_1(:,1,4),chip_size,chip_size);
imagesc([resol resol*chip_size],[resol resol*chip_size],Im1)
title('Scenario 1, 3dB SNR')
xlabel('km')
ylabel('km')

%Plot DR, PR, and FA
SNR = [20 10 5 3];

figure(2)
subplot(2,2,1)
plot(SNR, DR)
title('Detection Rate vs SNR')
xlabel('SNR (dB)')
ylabel('Detection Rate (%)')
subplot(2,2,2)
plot(SNR, PR);
title('Precision Rate vs SNR')
xlabel('SNR (dB)')
ylabel('Precision Rate (%)')
subplot(2,2,3)
```

```
plot(SNR, FA);
title('False Alarms vs SNR')
xlabel('SNR (dB)')
ylabel('False Alarms')


figure(5)
%imagesc(reshape(Fg_chip_hist(:,1),150,150))
 for i = 1:nt
    x1 = tracks(i).histxp(:,1)/10;
    y1 = tracks(i).histxp(:,3)/10;
    x2 = tracks(i).histxm(:,1)/10;
    y2 = tracks(i).histxm(:,3)/10;

plot(x1,y1,'-' , 'color', rand(1,3)); hold on
plot(x2,y2,'*' ,'color', rand(1,3)); hold on
axis([0 15 0 15])
title('Kalman Filter Tracks')
xlabel('km')
ylabel('km')
legend('KF Predicted','Measurement')
 end



function [ LCM_Image ] = LCM_enhance( Image_chip )
%-----------------------------------------------------------------------
----
% LCM_enhance.m
% 11/23/14
%
% This funciton imports a 150 x 150 image chip as a vector and applies
the LCM
% algorithm.
%
% Input:
%   Image_chip = m x n matrix
%                m = vector of pixels that make up image (Intensity
Image
%                0–1)
%                n = image frame
% Output:
%   LCM_Image = m x n matrix conveted to vector
%-----------------------------------------------------------------------
----

A = Image_chip;
chip_size = sqrt(size(A,1));
n=size(A,2); % Number of Iterations

for pp = 1:n

L(:,:,1) = reshape(A(:,pp),chip_size,chip_size);
L(:,:,2)= shiftu(L(:,:,1),0,1);
```

```
L(:,:,3)= shiftd(L(:,:,1),0,1);
L(:,:,4) = shiftl(L(:,:,1),0,1);
L(:,:,5) = shiftr(L(:,:,1),0,1);
L(:,:,6) = shiftl(L(:,:,2),0,1);
L(:,:,7) = shiftr(L(:,:,2),0,1);
L(:,:,8) = shiftl(L(:,:,3),0,1);
L(:,:,9) = shiftr(L(:,:,3),0,1);

N = (ones(chip_size,chip_size));
N = N*9;
N(1,1)=4;
N(end,end) = 4;
N(1,end) = 4;
N(end,1) = 4;
N(2:end-1,1)=6;
N(1, 2:end-1) = 6;
N(2:end-1,end)=6;
N(end, 2:end-1) = 6;

A_mean = (sum(L,3))./N;
L = max(L,[],3);
L2=L.^2;

su = shiftu(A_mean,0,n,1);
sd = shiftd(A_mean,0,n,1);
sl = shiftl(A_mean,0,n,1);
sr = shiftr(A_mean,0,n,1);
susl = shiftl(su,0,n,1);
susr = shiftr(su,0,n,1);
sdsl = shiftl(sd,0,n,1);
sdsr = shiftr(sd,0,n,1);

A_mean_3d(:,:,1) = L2./A_mean; clear A_mean;
A_mean_3d(:,:,2) = L2./su; clear su;
A_mean_3d(:,:,3) = L2./sd; clear sd;
A_mean_3d(:,:,4) = L2./sl; clear sl;
A_mean_3d(:,:,5) = L2./sr; clear sr;
A_mean_3d(:,:,6) = L2./susl; clear susl;
A_mean_3d(:,:,7) = L2./susr; clear susr;
A_mean_3d(:,:,8) = L2./sdsl; clear sdsl;
A_mean_3d(:,:,9) = L2./sdsr; clear sdsr;

C_min = min(A_mean_3d,[],3);
C_min=mat2gray(C_min);
LCM_Image(:,pp) = C_min(:);
clear A_mean_3d

end



function [ Fg_hist ] = PCA_Detect(image_chip,w,LCM)
%-----------------------------------------------------------------------
----
```

```
% PCA.m
% Principle Component Analysis
% 11/1/2014
%
% Inputs:
% - image_chip (150 x 150) matrix
% - w = sliding window for PCA
% - LCM = 0 or 1, use to include of exclude LCM
%
% This script uses a principle component analysis to determine the
% background an then subtract it from the current scene.
%
% Charles Guyon, Thierry Bouwmans and El-hadi Zahzah, Robust Principle
% Component Analysis for Background Subtraction: Systematic Evaluation
and
% Comparative Results, Lab. MIA - Univ. La Rochelle France, downloaded
from
% www.intechopen.com on 31 Oct 2014
%-----------------------------------------------------------------------
----

%load('Image_hist.mat')
M = image_chip; % Data Set
n = size(M,2);  % Number of frames
chip = sqrt(size(M,1));

for i = w:n


A = M(:,i+1-w:i);  % Build Window


[U, S, V] = svds(A);
%U = pca(A);

Bg = U(:,1:2)*U(:,1:2)'*M(:,i);  % Background
Bg = Bg./max(Bg);            % Normalize Background

% Normalize background to scene
Fg = abs(M(:,i)-Bg);

if LCM==1
    Fg_LCM = LCM_enhance(Fg);
    Fg_hist(:,i+1-w) = Fg_LCM;
else
    Fg_hist(:,i+1-w) = Fg;
end


end
end
```

```
function [ Target_Detects ] = Target_Detection( Image_chip ,k,min_l,
max_l )
%-----------------------------------------------------------------------
----
% Target_Detection.m
% 11/21/14
%
% This funciton detects targets within sensor frames using the
threshold
% method below.
%
% Input:
%   Image_chip = m x n matrix
%                m = vector of pixels that make up image (Intensity
Image
%                0–1), note that the image chip is a square matrix when
%                reshaped.
%                n = image frame
%   k = gain used in threshold detection
%   min_l = minimum target length
%   max_l = maximum target length
% Output:
%   Target_Detects = Boundary Box Coordinates
%
% T = mean2(Image)+k*std(Image)
%-----------------------------------------------------------------------
----

n = size(Image_chip,2);  % Number of frames
m = sqrt(size(Image_chip,1)); %

for i = 1:n
    temp_image = reshape(Image_chip(:,i),m,m);
    Thresh = mean2(temp_image)+k*std2(temp_image);
    temp_bw = im2bw(temp_image,Thresh);
    STATS = regionprops(temp_bw, 'BoundingBox'); %Performs shape
measurements
    sl=size(STATS,1);



    TD = [];
    pp = 1;
    for p = 1 : sl
        BB = STATS(p).BoundingBox;
            if  (STATS(p).BoundingBox(3)<max_l &&
STATS(p).BoundingBox(4)<max_l &&STATS(p).BoundingBox(3)>min_l &&
STATS(p).BoundingBox(4)>min_l);
        TD(pp,:)= BB;
        pp=pp+1;
            end
    end
Target_Detects{:,:,i}=TD;
end
```

```
end



function J =Attack_trajectory(tf)

global a1 v0 vf vmin vmax psidotmax p0 p0ship psiShip psidot_max v_min
v_max vShip a0 af amax amin a_max a_min

% compute initial conditions
J=0;
[n m]=size(p0);
for k=1:m
    x0=p0(:,k);

% point the attacker at the ship

psi0 = atan2(p0(2)-p0ship(2),p0(1)-p0ship(1))-pi;

x0p = [cos(psi0);sin(psi0)];
x0pp=a0*[-sin(psi0);cos(psi0)];

%define speed profile

     lambda0 = v0;
     lambdaf = vf;

     tauf=lambda0*tf;

     c1 = (lambdaf-lambda0)/tauf;
     c2 = lambda0;

% compute tauf  %tf

    if lambda0 == lambdaf
         %tf = tauf/c2;
         tauf=c2*tf;
    else
         %tf = log((c1*tauf + c2)/c2)/c1;
         tauf=(c2*exp(c1*tf)-c2)/c1;
    end

% compute final conditions

xf = p0ship + [vShip*cos(psiShip); vShip*sin(psiShip)]*tauf;

% determine the final heading at impact

perror = [cos(psiShip) sin(psiShip);-sin(psiShip)
cos(psiShip)]*(p0(:,k)-p0ship);
```

```
if  perror(2) >= 0
    psif = psiShip - pi/2;
else
    psif = psiShip + pi/2;
end

xfp = [cos(psif);sin(psif)];
xfpp=af*[-sin(psif);cos(psif)];
% evaluate path coefficients

a1 = coefficients(x0,x0p,x0pp,xf,xfp,xfpp,tf);

% compute max turn rate and max and min velocity using equations
derived in
% the notes

N = 50; dt = tf/N;
v_max = vmin;
v_min = vmax;
a_max = amin;
a_min = amax;
psidot_max = 0;
data = [];

for j = 0:N

     t = j*dt;

     % compute vehicle path and its derivatives
     p = zeros(2,1);
     for i = 1:6
         k = i-1;
         p = p + a1(:,i)*t^(k);
     end

     p_p = zeros(2,1);
     for i = 2:6
         k = i - 2;
        p_p = p_p + a1(:,i)*t^(k);
     end

     p_pp = zeros(2,1);
     for i = 3:6
         k = i - 3;
         p_pp = p_pp + a1(:,i)*t^(k);
     end

     % compute speed

     lambda = c1*t + c2;
     v = lambda*norm(p_p);
     % compute turn rate
     psi_dot = (p_pp(2)*p_p(1) - p_pp(1)*p_p(2))/norm(p_p)^2;

     % compute acceleration

     lambda_p=c1;
     a=norm(p_pp)*lambda+norm(p_p)*lambda_p*lambda;

     %compute max turn rate and max and min speed and acceleration
along the path

     psidot_max = max(psidot_max,psi_dot);
     v_max = max(v,v_max);
     v_min = min(v,v_max);
     a_max = max(a,a_max);
     a_min = min(a,a_max);

end

% the cost includes minimum time and weighted penalty on the terms that
% need to be linited. The weights are selected by the designer
end
 J = J+tf + 1*(psidot_max - psidotmax)^2/psidotmax^2 + (v_min -
vmin)^2/vmin^2 + 1e6*(v_max - vmax)^2/vmax^2+1e6*(a_max-amax)^2/amax^2;
 Return



function [tracks] = KF_Tracker2(tracks, delta)
% KF_Tracker.m
%
% xm - measured state
% xhat - predicted state
% kfinit - 0 if 1st iteration
% delta = time step

n = size(tracks,2);



for i = 1:n

    xhat = tracks(i).xp';
    xm = tracks(i).xm';
    P = tracks(i).P;
% Initilize Filter
I = eye(4);

R = [.1 .0001;
    .0001 .1];

F=[1 delta 0 0;
    0 1 0 0;
    0 0 1 delta;
    0 0 0 1];

omega=[delta^3/3;delta;delta^3/3;delta]; % plant covariance

q=0;                     % No Plant Noise for 1st Simulation

Q=q^2*[delta^3/3 delta^2/2 0 0;
    delta^2/2 delta 0 0;
    0 0 delta^3/3 delta^2/2;
    0 0  delta^2/2 delta];

% Define Measurement Matrix, H
H=[1 0 0 0; 0 0 1 0];

% Prediction Step

xhat=F*xhat; % state prediction, xhat(k+1|k)
P=F*P*F'+Q;  % covariance prediction, P(k+1|k)

% Process measurement from Sensor 1
z = H*xm;

% Correction Step

z_tilda=z-H*xhat(:,end);   %difference between measurement and
prediction
K=P*H'*inv(H*P*H'+R); % Compute Kalman Gain
xhat=xhat(:,end)+K*z_tilda; % Correct state, xhat(k+1|k+1)
P=(I-K*H)*P*(I-K*H)'+K*R*K'; % Update Covaraiance P(k+1|k+1)

tracks(i).xp = xhat';
tracks(i).P = P;
tracks(i).histxp = [tracks(i).histxp; xhat'];
end

end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] B. N. Agrawal, “Detection and Tracking Algorithms for OPIR Optics,” Naval Postgraduate School research proposal, 2013, unpublished.

[2] Los Angeles Air Force Base. (2014, Jan. 7). “*Infrared Space Systems Directorate*” [Online]. Available: http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=5514 [Dec. 08, 2014].

[3] B. N. Agrawal, “Integration and test of HEL Beam Control Research Testbed adaptive optics system for solid state laser,” Naval Postgraduate School research proposal, 2013, unpublished.

[4] C. Kopp, “High energy laser directed energy weapons,” APA, Tech. Rep. APA-TR-2008–0501, Air Power Australia, Apr. 2012.

[5] High-energy laser. (n.d.). Office of Naval Research Science and Technology. [Online]. Available: http://www.onr.navy.mil/en/Media-Center/Fact-Sheets/High-Energy-Laser.aspx. Accessed Oct. 14, 2014.

[6] R. Scott. (2014). “Laser weapon breaks cover on USS Ponce,” in *IHS Jane’s Navy International* [Online]. London: HIS Jane’s 360. Available: http://www.janes.com/article/46126/laser-weapon-breaks-cover-on-uss-ponce [Dec. 08, 2014].

[7] K. Seyrafi and S. A. Hovanessian, *Introduction to Electro-Optical Imaging and Tracking Systems*. Norwood, MA: Artech House, Inc., 1993.

[8] P. Merritt, *Beam Control for Laser Systems*, 1st ed. Albuquerque, NM: The Directed Energy Professional Society, 2012, pp. 189–208.

[9] Frequently asked questions about the landsat missions. (n.d.). United States Geological Society. [Online]. Available: http://landsat.usgs.gov/band_designations_landsat_satellites.php. Accessed Oct. 14, 2014.

[10] L. J. Austin III. (2014, Mar. 5). USCENTCOM Commander’s posture statement before the House Armed Services Committee. [Online]. Available: http://www.centcom.mil/en/about-centcom-en/commanders-posture-statement-en.

[11] A. Tiwari, “Small boat and swarm defense: A gap study,” M.S. thesis, Dept. of Comput. Sci, NPS, Monterey, CA, 2008.

[12] C.L. Chen, H. Li, Y. Wei, T. Xia, and Y. Y. Tang, “A Local Contrast Method for Small Infrared Target Detection,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 574–581, Jan. 2014.

[13] D. Liu and J. Yu, “Otsu method and K-means,” in *2009 9th Int. Conf. on Hybrid Intelligent Systems,* Beijing Jiaotong University, Beijing, China, 2009, pp. 344–349.

[14] C. Guyon, T. Bouwmans, and E. Zahzah, “Robust Principal Component Analysis for Background Subtraction: Systematic Evaluation and Comparative Analysis,” in *Robust Principal Component Analysis for Background Subtraction: Systematic Evaluation and Comparative Analysis*, La Rochelle, France: MIA Lab. Univ. La Rochelle, 2012, ch. 12, pp. 223–238.

[15] G. Koretsky, J. Nicoll, and M. Taylor, “A Tutorial on Electro-Optical/Infrared (EO/IR) Theory and Systems,” Institute for Defense Analysis, IDA Doc. D-4642, Alexandria, VA, Jan. 2013.

[16] M. Alleva, private communication, Boeing DES, Oct. 2014.

[17] D.M. Hussain, D. Hicks, D. Ortiz-Arroyo, S. Haq and Z. Ahmed, “A Case Study: Kalman & Alpha-Beta Computation under High Correlation,” *IMECS*, vol. I, Hong Kong, China, Mar. 2008.

[18] M. Borges (2013), *alphaBetaFilter.m*, (Version v1), [MATLAB code]. Available: http://www.mathworks.com/matlabcentral/fileexchange/42223-alphabetafilter/content/alphaBetaFilter.m. Accessed Sep. 17, 2014.

[19] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis, and G. Karaseitanidis, “Multiple Hypothesis Tracking,” in *Laser Scanner Technology,* Dr. J. A. Rodriguez (Ed.), Rijeka, Croatia: InTech, 2012, ch. 10, pp. 199–220, Available from: http://www.intechopen.com/books/laser-scanner-technology/multiple-hypothesis-tracking-implementation

[20] T. J. Ma, private communication, Sandia National Laboratories, Oct. 2014.

[21] M. Dragovic, “Velocity Filtering for Target Detection and Track Initiation,” Defense Science and Technology Organization, Edinburgh, Australia, Tech. Rep. DSTO-TR-1406, Mar. 2003.

[22] G. Mills-Tettey, A. Stentz, and M. Dias, “The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs,” Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07–27, Jul. 2007.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California